RAS++: REPRESENTING HYBRID REUSE ASSETS FOR MDE AS A SERVICE

Fábio Paulo Basso

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientador: Toacy Cavalcante de Oliveira

Rio de Janeiro
Setembro de 2017

RAS++: REPRESENTING HYBRID REUSE ASSETS FOR MDE AS A SERVICE

Fábio Paulo Basso

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2017

Basso, Fábio Paulo

RAS++: Representing Hybrid Reuse Assets for MDE as a Service/Fábio Paulo Basso. – Rio de Janeiro: UFRJ/COPPE, 2017.

XVI, 239 p.: il.; 29, 7cm.

Orientador: Toacy Cavalcante de Oliveira

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2017.

Referências Bibliográficas: p. 198–228.

1. Model-Driven Engineering. 2. Reusable Asset. 3. Pivot Language. 4. MDE as a Service. 5. MDE Settings. I. Oliveira, Toacy Cavalcante de. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Para Raquel.

Agradecimentos

Em primeiro lugar, à minha família pelo apoio, amor e incentivo neste período em que estive distante. Em especial à minha esposa, Raquel, que sempre me deu apoio e foi crucial para a conclusão desse trabalho. À minha mãe, Maria Beatriz Basso, e meu pai, Irani Paulo Basso, pela eterna preocupação com meu bem-estar mesmo de longe e às minhas irmãs Paula e Cíntia.

Ao meu orientador, professor Toacy Cavalcante Oliveira, que sempre esteve presente e comprometido com a qualidade desse trabalho.

À professora Cláudia Werner, que foi a maior incentivadora e contribuidora dessa pesquisa. Obrigado por me auxiliar na condução da tese, dedicando tempo para reuniões e revisões de material escrito, que por fim resultaram nessa tese de doutorado.

Aos professores Leandro Buss Becker, Rafael Zancan Frantz, Fabrícia Rooz-Frantz e Kleinner Farias por contribuirem na execução de estudos e divulgação dos resultados desse trabalho.

Ao CNPq e CAPES, pelo apoio financeiro (processo 141792/2014-0) durante o doutorado.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

RAS++: REPRESENTANDO ATIVOS DE REÚSO HÍBRIDOS PARA MDE COMO UM SERVIÇO

Fábio Paulo Basso

Setembro/2017

Orientador: Toacy Cavalcante de Oliveira

Programa: Engenharia de Sistemas e Computação

Artefatos relacionados à Engenharia Dirigida por Modelos (MDE), tais como transformações de modelo, Linguagens Específicas de Domínio (DSLs) e ferramentas de modelagem ou refinamento, têm sido propostos na literatura visando aumentar a qualidade de produtos derivados de atividades da Engenharia de Software. Estes artefatos são introduzidos em configurações de nível técnico, incluindo DSLs e outras formas para se representar cadeias de ferramentas. Uma introdução bem sucedida de MDE em contextos alvos inclui a realização de fases de integração, que estabelecem cadeias de ferramentas customizadas. Esta customização tem sido realizada por engenheiros de software no então chamado "MDE como um Serviço", onde novas oportunidades para o estabelecimento de cadeias de ferramentas estão disponíveis em repositórios de ativos por meio de cenários de coopetição (colaboração entre empresas competidoras). Coopetição beneficia líderes de um mercado assim como seus competidores, podendo auxiliar na promoção do MDE em um futuro próximo. Para tanto, é necessária uma representação comum/híbrida para ativos e cadeias de ferramentas, o que representa uma limitação no estado da arte atual. Ao incluir propriedades híbridas, uma representação comum simplificaria a integração de cenários para coopetição no MDE, permitindo a transformação automática de uma característica estrutural de um cenário para outro. Assim, essa tese propõe RAS++, uma nova linguagem de representação para ativos híbridos. Os resultados dessa pesquisa indicam que RAS++ é representativa o suficiente para apoiar implementações de "MDE como um Serviço" nos cenários de coopetição avaliados.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

RAS++: REPRESENTING HYBRID REUSE ASSETS FOR MDE AS A SERVICE

Fábio Paulo Basso

September/2017

Advisor: Toacy Cavalcante de Oliveira

Department: Computer Science and Systems Engineering

Artifacts associated with Model-Driven Engineering (MDE) such as model transformations, Domain Specific Languages (DSL), and modeling or refinement tools have been proposed in the literature as a mean to increase the quality in products derived from activities of Software Engineering. These artifacts are introduced in technical-level settings, including DSLs adopted by model transformation engines, software project workspaces, and other ways to represent tool chains. In technical terms, a successful MDE introduction in target contexts includes the execution of integration phases that establish customized tool chains. This customization has been performed by software engineers in the called "MDE as a Service", where new opportunities for tool chain are available in asset repositories through coopetition scenarios (collaboration between competing companies). Coopetition benefits market leaders and their competitors and may help promoting MDE adoption. This way, it is necessary a common/hybrid representation for assets and tool chain, which represents a limitation in the state of the art. By including properties from MDE Artifact repositories and tool chain representations, a common representation would simplify the integration of scenarios for coopetition in MDE, allowing automatic transformation of structural features from a scenario to another one. Thus, we proposed RAS++, a new representation language for hybrid assets. Our research results indicate that RAS++ is representative enough to support implementations of MDE as a Service in evaluated coopetition scenarios.

Index

List of Figures

List of Tables

Chapter 1

Introduction

> The only sure way to avoid making mistakes is to have no new ideas.
>
> Albert Einstein

1.1 Context

Software systems are becoming more complex. The number of functionalities requested by clients has increased as well as the complexity to develop them; systems are built from other systems or including legacy code and are distributed over diverse sites and platforms. These challenges require specific approaches, which often involve domain-specific knowledge and solutions. However, based on the experience obtained from several domains and projects, some solutions may bring benefits to the complex software development.

MDE is an approach built on widespread techniques in Software Engineering, which presents the following main characteristics: (1) uses models in all the phases of software development to improve understanding; (2) raises the abstraction level of software system specifications, hiding platform-specific details; (3) develops Domain-Specific Languages (DSLs) and frameworks to suit a domain; and (4) applies transformations to automate repetitive activities and improve product quality derived from Software Engineering (e.g., source code, libraries, processes, etc.) (MOHAGHEGHI, 2008). In other words, MDE proposes representation of diverse Software Engineering's artifacts through modeling and tool support.

MDE is sometimes used as synonymous of Model-Driven Development (MDD) (KÖNEMANN *et al.*, 2009; MONTEIRO *et al.*, 2014b; SELIC, 2006; TORCHIANO *et al.*, 2013). However, while MDD takes advantage of models to code generation, MDE uses models in diverse contexts and to different intents in Software Engineering (SOMMERVILLE, 2010). For example, MDE includes practical appli-

cations for system engineering (BECKER *et al.*, 2002; SELIC and RUMBAUGH, 1998; SELIC, 2005) and method engineering (BECKER *et al.*, 2007; HEBIG and BENDRAOU, 2014; MACIEL *et al.*, 2013). In this thesis, the term MDE is adopted to refer to mechanisms, techniques, or approaches using models and automation.

Additionally, MDE has been used to support software systems development in both academia (ARANEGA *et al.*, 2012a; BATORY *et al.*, 2013b; LUCAS *et al.*, 2017; MACIEL *et al.*, 2013) and industry (CAPILLA *et al.*, 2014; HUTCHINSON *et al.*, 2011; LIEBEL *et al.*, 2014; MOHAGHEGHI *et al.*, 2009, 2013; MONTEIRO *et al.*, 2014b; TORCHIANO *et al.*, 2013; WHITTLE *et al.*, 2015). Among the benefits credited to MDE, MOHAGHEGHI and DEHLEN (2008) highlight the following: Shortened development time and increased productivity; improved software quality; automation through generation of code and other artifacts; provision of a common framework for software development across the company and lifecycle phases; maintenance and evolution; improved communication and information sharing among stakeholders. MDE is especially interesting for scenarios involving systems that should be made available on multiple platforms (SELIC, 2005, 2014). MOHAGHEGHI *et al.* (2013) go beyond and also expose the criteria that led companies to adopt MDE. Such criteria involve the abstraction level that hides details, communicating with non-technical staff, as well as model-based simulation, execution, and test.

However, developing software systems with MDE is not trivial. It requires an automated process, including transformation scripts that connects MDE resources, such as refinement tools (BATORY *et al.*, 2013b) and model-based operations (ROSE *et al.*, 2013). MOHAGHEGHI *et al.* (2013) claim that required expertise and high costs affecting the development of this automated process represent an issue for MDE adoption, making it unfeasible for small contexts of software development.

MDE introduction in a target context involves integration phases to establish a customized tool chain (MOHAGHEGHI *et al.*, 2013) and the use of some reuse tools (KUSEL *et al.*, 2015; OLIVEIRA *et al.*, 2011), which instantiate this chain for specific needs. In addition, MDE adoption involves end-users contexts (WHITTLE *et al.*, 2015) as well as inter-organizational contexts (BOSCH, 2009). In this way, the creation or instantiation of tool chains implies in connecting two systems (endpoints) to interoperate (AHO *et al.*, 2009; BIEHL *et al.*, 2014; BRUNELIÈRE *et al.*, 2010; ELAASAR and NEAL, 2013; ZHANG, 2015).

Since interoperability ranges many aspects in Software Engineering (BIEHL *et al.*, 2014; FRANTZ and CORCHUELO, 2012; MOTTA *et al.*, 2017; YIE *et al.*, 2012; ZHANG and MOLLER-PEDERSEN, 2013), there is no common definition restricting the use of the term Tool Chain. For example, it has been discussed

as synonymous for Model Transformation Chain (MTC) (ETIEN *et al.*, 2013). In order to clearly set up the scope under investigation, we adopted the definitions from (HEBIG, 2014) and (ZAKHEIM, 2017) as follows.

According to HEBIG (2014), tool chain is an MDE Setting, a term defined as:

> *"the set of manual, semi-automated, and automated activities that are employed during development, the set of artifacts that are consumed or produced by these activities, the set of languages used to describe the artifacts, as well as the set of tools that allow the editing of used languages or that implement automated activities."*

According to ZAKHEIM (2017), establishing tool chains is:

> *"not a purely technical challenge ... In fact, it's more of a business problem. While there are a couple of choices one can make in selecting the technical integration infrastructure (integration via APIs or at the database layer), the real challenges have more to do with the friction caused by the dissimilarities among these tools and how to overcome them in order to effect a flawless integration."*

In this thesis, we embraced the aforementioned definitions to base our proposal. Next, we discuss on the motivation of the research.

1.2 Motivation

Software engineers know how to connect MDE resources. For example, the state of the practice has integrated tools for MDE (MOHAGHEGHI *et al.*, 2013; TORCHIANO *et al.*, 2013), connecting model transformations (LÚCIO *et al.*, 2014), as well as tools built on Domain Specific Modeling Languages (DSMLs) (KELLY and TOLVANEN, 2008) or Domain Specific Languages (DSLs) (VOELTER, 2009). Tool chains are also available in MDD-based processes (KÖNEMANN *et al.*, 2009; MONTEIRO *et al.*, 2014b), mapping to code features found in underlying implementation frameworks (KURTEV *et al.*, 2006). There are also ad-hoc approaches for tool chains (AHO *et al.*, 2009), developing connectors in Java to support tasks of automated software design (BATORY *et al.*, 2013a,b).

WHITTLE *et al.* (2015) claim that software companies adopting MDE request customizations in processes and tools. This implies in delivering instantiated tool chain artifacts, which include the generation of some format adopted for execution, such as XML files with process definitions (LUCAS *et al.*, 2017), configuration files supporting execution of model transformation chains (ARANEGA *et al.*, 2012a),

service specifications for tools on the web (BIEHL *et al.*, 2014), and model transformations (KUSEL *et al.*, 2015).

In order to add flexibility for tool chain instantiation in target contexts, the state of the art proposed integration of reuse concepts in tool chain representation languages. DSLs supporting model transformation chains allow the representation of execution chains and compositions (VANHOOFF *et al.*, 2006), which must be well-formed in valid sequences (YIE *et al.*, 2012). Well-known concepts related to tool chain include model weaving (JOUAULT *et al.*, 2010), Software Product Lines (SPL) (ARANEGA *et al.*, 2012a), Component-Based Development (CBD) (VALE *et al.*, 2016), and tool support built on principles of Generative Programming (GP) (BASSO *et al.*, 2013a).

Currently, MDE Artifacts are tied to representations associated with architectures, platforms, and processes (HEBIG and BENDRAOU, 2014). According to FRANCE *et al.* (2006), MDE Artifacts are UML diagrams, drawings, Java programs, method descriptions, test cases, metamodels, toolboxes, system or process models, model transformations, DSLs or model refinement tools. In other words, MDE Artifacts include any resource found in a regular tool chain (ZHANG and MOLLER-PEDERSEN, 2013).

Among many problems that still surround tool chain (LIEBEL *et al.*, 2014), it is possible to highlight the lack of foundations involving preliminary phases to tool chain. The state of the practice knows how to integrate tools but misses foundations for analysis of artifacts/tools (SMOLANDER *et al.*, 2017; ZAKHEIM, 2017). For example, current theory and practice related to tool chain are limited when we consider implementation of an emergent scenario in which unfamiliar artifacts are requested for inclusion in inter-organizational contexts (BOSCH, 2013, 2009). This scenario requires representation of descriptive information for classification, decision making and assessment of MDE Artifacts before applying any tool chain mechanism (ZAKHEIM, 2017).

Thereby, aiming at transforming implicit knowledge to explicit information, recent research has been motivating the representation of technicalities and descriptive information in assets through repositories for MDE Artifacts (COMBEMALE *et al.*, 2014; FRANCE *et al.*, 2007; GORP and GREFEN, 2012; JACOBSON *et al.*, 2012; ROCCO *et al.*, 2015). An asset is a data pack describing artifacts shared in repositories (AMS, 2014; OMG, 2005). According to modern ecosystem repositories (AXELSSON *et al.*, 2014; BADAMPUDI *et al.*, 2016; SANTOS *et al.*, 2016), assets characterize a possible solution for enabling cooperation and competition, a combination that has been referred as *coopetition* in business scenarios (RITALA *et al.*, 2014). In this sense, assets need to be discovered, understood and compared with other assets.

Coopetition has been implemented by companies like Amazon (PALMQUIST, 2014) and represents a long term goal in MDE research (COMBEMALE *et al.*, 2014; GORP and GREFEN, 2012; ROCCO *et al.*, 2015). RITALA *et al.* (2014) define a scenario for coopetition as:

> *Coopetition (collaboration between competing firms) is a phenomenon that has recently captured a great deal of attention due to its increasing relevance to business practice.*

The interest on assets as means to promote coopetition may be observed in the literature through reports related to integration services (ZAKHEIM, 2017), and through software companies that are building their portfolios on modern asset technologies. Currently, 91 world wide organizations[1] supervise and develop asset specifications based on Open Service for Lifecycle Collaboration (OSLC) (ELAASAR and NEAL, 2013; ZHANG and MOLLER-PEDERSEN, 2013), a specification built on asset concepts (BASSO *et al.*, 2016c).

Some companies have implemented coopetition through common representations for their business models built on services. Meanwhile, the MDE community has not still done the same. Despite the advances already achieved by the MDE community, there are still many open questions (MUSSBACHER *et al.*, 2014). Some contributions pointed out contextual, technical, and educational issues that hamper MDE in practice (PETRE, 2013; WHITTLE *et al.*, 2015). However, this scenario surrounding services for MDE is not well understood, needing characterization.

1.3 Problem Definition

As stated in Section 1.2, implementations of services for MDE are still not well understood and characterized, thus presenting the following issues.

Software engineering services for MDE are of interest of a community from the view point of business (LÚCIO *et al.*, 2014; MONTEIRO *et al.*, 2014a). Such services could introduce MDE resources in target contexts through reuse techniques. In this way, a given resource could be applied in the context of different companies, characterizing an inter-organizational reuse scenario. However, a common sense from industrial surveys and reports in the area (MONTEIRO *et al.*, 2014a; WHITTLE *et al.*, 2015) is that MDE resources require customization before being introduced in a target context. KUSEL *et al.* (2015) concluded from a systematic literature review that reuse in the context of MDE research is still immature, notably lacking relevant and evaluated approaches.

[1]OSLC - <http://open-services.net/organizations/> (last access on September 1st, 2017)

In order to improve understanding on services for MDE, important experiences from the past should not be ignored (MONTEIRO *et al.*, 2014a; TORCHIANO *et al.*, 2013). According to MOHAGHEGHI *et al.* (2013), past experiences on introducing MDE in target contexts should be reported through longitudinal studies (RUNESON and HÖST, 2008). These studies would allow obtaining a better understanding of this scenario and to prospect improvements in the future. Therefore, the body of knowledge on services for MDE needs more studies and related contributions are welcome (MUSSBACHER *et al.*, 2014).

Among many reasons why coopetition is not a reality in MDE business (JAKUMEIT *et al.*, 2014), the lack of a common representation deserves attention (BASSO *et al.*, 2015a; BASSO, 2016). However, it is not clear in the literature which information should be represented in the context of tool chain and assets (BASSO *et al.*, 2017a). These different contexts present structural features that are still not integrated in existing representation languages for assets or chains, characterizing a limitation in the state of the art that hampers integration initiatives (SMOLANDER *et al.*, 2017; ZAKHEIM, 2017).

According to RITALA *et al.* (2014), coopetition through services requires a common language. Then, a good foundation starts by finding common properties to represent assets and tool chain (JAKUMEIT *et al.*, 2014). Since assets need to be discovered, understood and compared with other assets, this common representation should consider properties from two different worlds: repositories and tool chain. Thus, another issue is the lack of mapping studies discussing properties from assets that should be used in services for MDE.

Currently, there is no common representation language for services in MDE. Such a language should increment the state of the art with more representative and non-ambiguous concepts in structural features of assets. This is also a challenge and requires mappings of existing contributions in the area with a grouping study, e.g., following the protocol defined by RUNESON and HÖST (2008).

As consequence, the current scenario is unconnected, making hard, if not impossible, the implementation of coopetition for services in MDE. For example, currently software engineers may use at least five options of repositories supporting assets for MDE Artifacts (COMBEMALE *et al.*, 2014; FRANCE *et al.*, 2007; GORP and GREFEN, 2012; JACOBSON *et al.*, 2012; ROCCO *et al.*, 2015), all of them adopting different types of asset representations. Moreover, in an integration process, artifacts from selected assets are manually inserted in a tool chain, which has at least 60 different representation options. Therefore, due to the expressive number of integration possibilities, another issue related to implementation of coopetition in MDE is the high cost involved to develop transformations/connectors (MOHAGHEGHI *et al.*, 2013).

Finally, this diversity (or hybridization) of asset representations implies in great manual effort for integration: A quadratic combinatorial explosion for converting different representations (BASSO, 2015).

1.4 Research Rationale

Goals for this research are derived from the problems stated in Section 1.3 and are structured in general and specific ones. Our general research goal consists on proposing a common representation language for assets in the context of MDE, as well as contributing with studies of characterization on services for MDE, which we have called "MDE as a Service". Next, we present our specific research goals.

1.4.1 Research Goals

Goal 1: Enhancing knowledge about MDE as a Service in intra and inter-organizational contexts. We intent to contribute to such goal by building foundations for this complex reuse scenario under the term "MDE as a Service" (or MDEaaS), as illustrates Figure 1.1. This way, software engineers visualize business opportunities for creating tool chains (region 2 of the figure) and instantiating them for specific requests from customers (region 3). In order to improve understanding on services for MDE, we will report our past experiences from the industry on introducing MDE in target contexts through longitudinal studies, as recommended by RUNESON and HÖST (2008). Moreover, we will present new research to identify relevant aspects related to implementations of MDE as a Service. Thus, our studies may contribute to clarify this research area still poorly understood.

Goal 2: Providing analysis of tool chain and asset representations. To this objective, we intent to analyze MDE toolboxes supporting tool chain in MDE as a Service scenarios as well as repositories sharing MDE assets. As a result, we will have a mapping study linking existing contributions in the area to applicability issues in practice. In addition, we intent to identify relevant concepts used in tool support that help software engineers to introduce MDE Artifacts in target contexts through MDE Settings (HEBIG, 2014).

Goal 3: Providing a common representation language that enables to capture MDE Artifacts and Settings in asset specifications. In this direction, an open question is which properties should be included in this language? The lack of a classification for existing MDE Toolboxes supporting assets and tool chain makes difficult the comparison and discovery of properties in approaches. Besides, there are many toolboxes with equivalent concepts (ambiguity issue). The discovery of common properties would allow us to construct the representation language

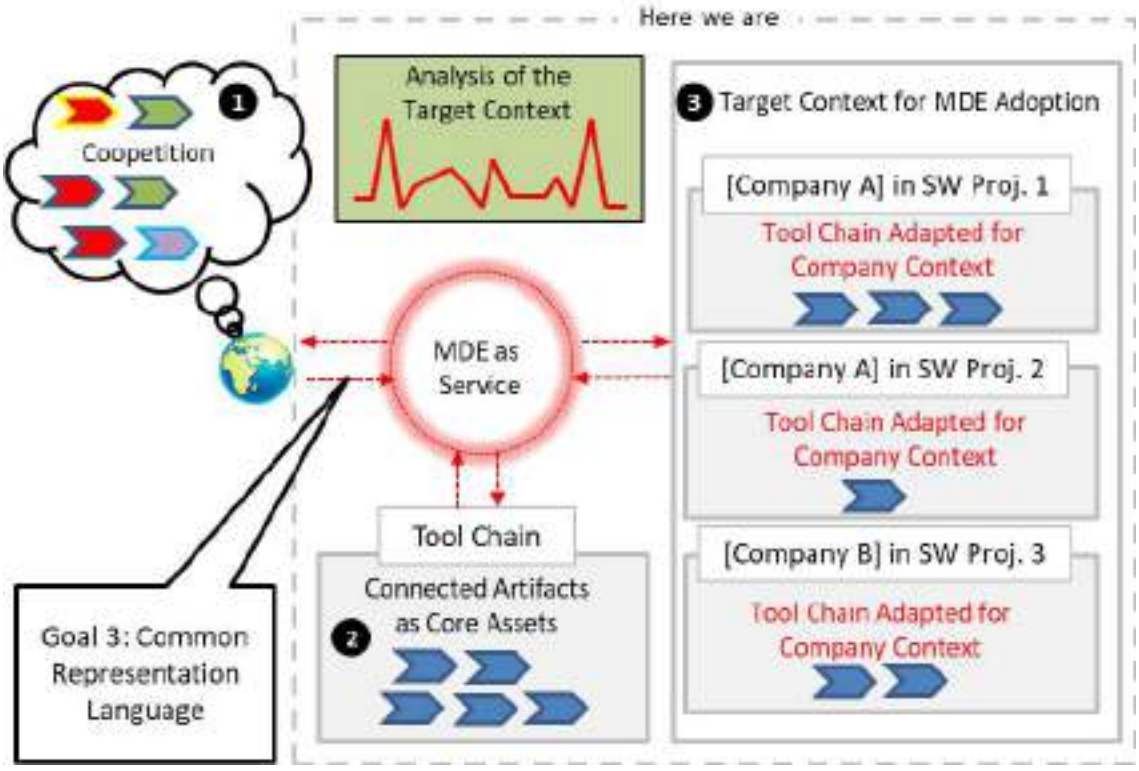


Figure 1.1: Emergent scenario for MDE as a Service

RAS++ without ambiguous metaclasses and properties. Currently, two specifications are available to represent assets: Reusable Asset Specification (RAS) (OMG, 2005), an OMG standard, and Asset Management Specification (AMS) (AMS, 2014), an OSLC standard. In this direction, because these specifications are standards, we advocate the need of a common representation language built on RAS and AMS properties.

Goal 4: Analyzing benefits and drawbacks related to RAS++ (our common representation language). Even though some studies compare one or other concept in the software engineering discipline, little has been done to create a broader view on options available to implement similar approaches to RAS++, using different concepts or properties (overlapping and complementarity issue). Thus, our goal here is to provide an analysis of benefits and drawbacks associated with RAS++ in a scenario of coopetition, including discussions related to services, assets, pivot (or pivotal) representations, and tool chain in MDE.

To this end, we have organized our research as described in the following sections.

1.4.2 Research Questions

The research questions for this thesis are derived from the problems mentioned in Section 1.3 and goals stated in Section 1.4.1, establishing directions and scope of

investigation.

Goal 1: Enhancing knowledge about MDE as a Service in intra and inter-organizational contexts.

- **Q1:** Which are the relevant aspects to be concerned with in implementations of MDE as a Service?

Goal 2: Providing analysis of tool chain and asset representations.

- **Q2:** Which are the relevant aspects to be concerned with in the integration of MDE Artifacts and Settings in the context of MDE as a Service?
- **Q3:** Which are the opportunities for coopetition in the context of MDE as a Service?
- **Q4:** Which studies should we select for evaluation of a common representation language in the context of MDE as a Service?

Goal 3: Providing a common representation language that enables to capture MDE Artifacts and Settings in asset specifications.

- **Q5:** Which are the properties required in RAS++ in the context of MDE as a Service for asset representation?
- **Q6:** Is RAS++ representative to play the role of a common language for the assets available in the ReMoDD repository?

Goal 4: Analyzing benefits and drawbacks related to RAS++ (our common representation language).

- **Q7:** Which is the effort required to implement coopetition in the context of MDE as a Service?
- **Q8:** Is RAS++ representative to play the role of a common language for tool chain and assets in the context of MDE as a Service?
- **Q9:** Why adopt RAS++ in future implementations of MDE as a Service?

These research questions are concerned with conceptual foundations for representation of information rather than tool support and implementation of business strategies for MDE as a Service. The aspects under investigation are longitudinal studies of characterization, analysis of properties and case studies for evaluation of representativeness. In the next section, we describe our research methodology and our objectives as milestones.

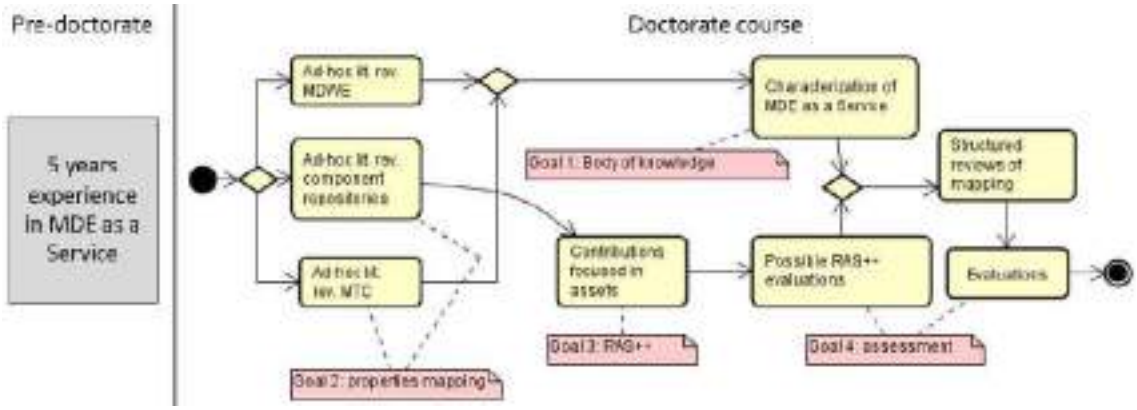


Figure 1.2: Application of our research methodology

1.4.3 Research Methodology

We adapted the methodology suggested by PEFFERS *et al.* (2007), starting with an initial investigation about problem definition (Section 1.3), research goals (Section 1.4.1) and research questions (Section 1.4.2). Figure 1.2 presents the stages composing our research methodology, where each activity is a complete execution of the Peffers's method composed of: 1) Identify the problem and define the objectives of the solutions; 2) Design and development of the solution; 3) Implementation/demonstration; 4) Evaluation; and 5) Communication.

Our research idea emerged from disciplines of software reuse. In these disciples, some issues, such as the limitations of the Reusable Asset Specification (OMG, 2005) for representation of hybrid assets, the lack of platforms for collaboration and competition in MDE, limitations in reuse approaches supporting tool chain, and the need for a common representation language, were raised, originating in the outline of this research.

The initial investigation was conducted based on our previous experience in three main areas of knowledge:

1. An ad-hoc literature review on web application engineering approaches. The result was quite interesting because this research topic has already been implemented in previous initiative for MDE as a Service (BASSO *et al.*, 2012). We then mined a software repository, extracting data sets from three software projects on the development of information systems with MDE;

2. An ad-hoc literature review on component repositories and assets. Initially, research collaborators (Toacy and Cláudia) recommended an investigation of RAS standard (OMG, 2005), a specification language usually adopted to represent semantics to software components. After that, it was analyzed a global reuse scenario for MDE, where it was possible to extract characteristics of

assets and carry out the first studies based on toy examples (BASSO *et al.*, 2013b,c);

3. An ad-hoc literature review on reuse and instantiation of model transformation and tool chains. The result was interesting because this research topic has already been investigated in the author's Master's dissertation (BASSO *et al.*, 2006).

At this stage, a big set of technical contributions were consulted, as well as relevant state of the practice research or proposals. This material was helpful to establish the review's scope and comparison features discussed in each of our publications. Based on such information, we started a series of studies discussing innovations from our previous experiences in terms of tool support, methodology, research challenges and positions that could contribute to the body of knowledge in MDE.

After communicating our first proposal for assets (BASSO *et al.*, 2013b,c), we realized that should report challenges associated with previous industrial experiences on the implementation of MDE as a Service. We then analyzed and exposed these challenges in publications shown in Tables 1.1 and 1.2. These publications provide foundations for MDE as a Service considering evaluation perspectives described in Table 1.3. Except TC05 and TC06, that are new studies, the data set was generated between 2006 and 2011 in real scenarios of software projects. These projects were mined, extracting data sets through the Eclipse Metrics Plugin[2].

We achieved our first goal by characterizing MDE as a Service through contributions shown in Tables 1.4, 1.1 and, specially in 1.2, where we discussed issues related with collaboration. In parallel, in order to achieve Goals 2 and 3, we used the body of knowledge derived from previous studies to start our main contribution: the RAS++ DSL. Table 1.4 shows publications derived from our study focused in asset specification languages. Except AS06, other five contributions adopted weak assessments (see column 2).

In order to reduce this bias, we planned a controlled experiment (RUNESON and HÖST, 2008) to be executed in a real setting: the MDArte project (MONTEIRO *et al.*, 2014b). However, the project was discontinued and its teams dissolved. We did not find another group with the required technical knowledge working on real MDE as a Service projects, we agreed that it would not be worth to run this experiment with other focal group. Since then, we adopted a new evaluation strategy based on analytical studies for Mining Software Repositories (MSR) (D'AMBROS *et al.*, 2008) and structured mapping studies (RUNESON and HÖST, 2008). This strategy was considered adequate in four events shown in Table 1.5: A Doctoral Symposium, two Workshops and Student Research Competition. In these events, we discussed about

[2]Eclipse Metrics - <http://metrics.sourceforge.net/> (last access on September 1st, 2017)

evaluation goals and presented some preliminary results of RAS++, our proposal for a common representation language.

Table 1.1: Contributions focused on tool chain contexts

LEGEND [MSR = Mining Software Repository]

Id	Reference	Title	Where?
TC01	BASSO *et al.* (2012)	Towards a Web Modeling Environment for a Model Driven Engineering Approach	BW-MDD
TC02	BASSO *et al.* (2014d)	Assisted Tasks to Generate Pre-prototypes for Web Information Systems	ICEIS
TC03	BASSO *et al.* (2014f)	Study on Combining Model-Driven Engineering and Scrum to Produce Web Information Systems	ICEIS
TC04	BASSO *et al.* (2014e)	Generative Adaptation of Model Transformation Assets: Experiences, Lessons and Drawbacks	SAC
TC05	BASSO *et al.* (2014c)	Extending JUnit 4 with Java Annotations and Reflection to Test Variant Model Transformation Assets	SAC
TC06	PAULON *et al.* (2014)	Wireless Sensor Network UML Profile to Support Model-Driven Development	INDIN

Id	Assessment Type	Research Method	Artifact Dev. Contexts	Assessment Perspectives	Representation Languages
TC01	Experience Report	Descriptive	Industry	P03, P05, P08, P09	WCT
TC02	Case Study/Team	Descriptive	Industry	P02, P04, P09	MockupToME DSLs
TC03	Case Study/Team	Descriptive	Industry	P01, P03, P04	MockupToME DSLs
TC04	Case Study/MSR	Descriptive	Industry	P03, P04, P05, P06	FOMDA DSL
TC05	Benchmark Study	Improving	Academy	P02, P04, P05, P09	FOMDA DSL
TC06	Action research	Improving	Academy	P03, P05, P09	FOMDA DSL

Table 1.2: Characterization studies for MDE as a Service

Id	Reference	Title	Where?
CS01	BASSO *et al.* (2014a)	Supporting Large Scale Model Transformation Reuse	ACM SIGPLAN Notices
CS02	BASSO *et al.* (2015b)	Combining MDE and Scrum on the Rapid Prototyping of Web Information Systems	IJWET
CS03	BASSO *et al.* (2016d)	Automated Design of Multi-layered Web Information Systems	JSS
CS04	BASSO *et al.* (2017a)	Building the Foundations for "MDE as Service"	IET Software

Id	Assessment Type	Research Method	Artifact Dev. Contexts	Assessment Perspectives	Representation Languages
CS01	Longitudinal Study	Descriptive	Industry	P03, P05, P08, P09	FOMDA DSL
CS02	Case Study/Team	Explanatory	Industry	P01, P03, P04	MockupToME DSLs
CS03	Longitudinal Study	Explanatory	Industry & Academy	P01, P02, P03, P04, P08	MockupToME DSLs
CS04	Longitudinal Study	Explanatory	Industry & Academy	P03, P04, P05, P07, P08	FOMDA DSL

In order to answer questions 2-4 from Goal 2, we restructured our unique mapping study from the qualification exam, conducting five new structured mappings, shown in Table 1.6. Only the first of these mappings is included in this document. These mappings have not been submitted for publication since they are still under improvement or revision.

These mappings provided new data for analysis. Then, we applied a selective coding process (PANDIT, 1996) to obtain common properties for RAS++. In this way, we achieved partially the Goal 3 with answers for Q5 and Q6, discussed along the next chapters. To answer Q7, we conducted a case study on mining ReMoDD repository (FRANCE *et al.*, 2007). Finally, data extracted from these five mappings were also applied to answer research questions Q8-Q12 along this thesis, allowing us to achieve Goal 4.

Table 1.3: Assessment perspectives

Id	Assessment Perspective	Description
P01	Time-scale feasibility	Includes benchmark tests for tool chain, measuring the time-scale required by end-users to perform MDE-based tasks. The time-scale includes contexts found in primary studies considering feedback from teams while implementing some software requirement.
P02	Intra-Organizational Context	Evaluates whether a specific software project, developed in a unique company, presents issues for introduction of some MDE resource. These are primary studies presenting results not generalizable for different companies.
P03	Inter-Organizational Context	Considers how well a set of MDE Artifacts and Settings match needs from different company contexts. This study is also characterized by grouping a set of primary studies, building foundations for reuse among companies.
P04	Tool Chain Usage	Evaluates whether a certain sequence of integrated tools is adequate for requests from a specific context. These are primary or secondary studies evaluating performance of teams on the execution of integrated and automated software development tasks.
P05	Tool Usage	Considers how well a tool for design, refinement, transformation and tool chain instantiation, performs some tasks conducted by some MDE practitioner. These primary studies, therefore, consider isolated tools, differently from tool chain feasibility that considers them all integrated.
P06	Pivotal Feasibility	Includes analytical studies by comparing structural features from representation languages. These are primary studies and aim at evaluating the quality attribute "representativeness". It is considered an adequate perspective for evaluation of main contribution.
P07	Business Feasibility	Considers the economic feasibility for execution of specialized software engineering services for MDE, a research topic not clearly discussed by the literature. These are secondary studies whose knowledge is built on primary studies for MDE and entrepreneurship.
P08	Methodological Usage	Evaluates how well end-users accept some methodology based on MDE in a tool chain context. This includes primary studies considering different roles using a methodology in a running process. We focused in identifying learning-curve issues and improvement points in a method-level for tool chain.
P09	Reuse Mechanism Usage	Includes evaluation of reuse mechanism used on the development, adaptation and integration of MDE resources in tool chains. These are primary studies concerned in evaluations for MDE Resource customizations.
P10	Reuse Cost Characterization	This is recently considered. BECKER *et al.* (2007) state that it is possible to characterize reuse approaches with four cost dimensions: (1) complexity of the reuse situation, (2) repetition rate of the reuse situation, (3) cost of preparation, and (4) cost of utilization.

All these initiatives enabled us to evaluate the quality attribute for representativeness of RAS++ in support for future implementations for MDE as a Service.

Table 1.4: Contributions focused on assets

LEGEND [MSR = Mining Software Repository]

Id	Reference	Title	Where?
AS01	BASSO *et al.* (2013c)	How do You Execute Reuse Tasks Among Tools? A RAS Based Approach to Interoperate Reuse Assistants	SEKE
AS02	BASSO *et al.* (2013b)	A Common Representation for Reuse Assistants	ICSR
AS03	BASSO *et al.* (2014b)	Towards Facilities to Introduce Solutions for MDE in Development Environments with Reusable Assets	IRI
AS04	BASSO *et al.* (2016c)	Analysis of Asset Specification Languages for Representation of Descriptive Data from MDE Artifacts	CENTERIS
AS05	BASSO *et al.* (2017b)	Automated Approach for Asset Integration in Eclipse IDE	JSOS@ICSE
AS06	BASSO *et al.* (2017c)	Revisiting Criteria for Description of MDE Artifacts	JSOS@ICSE

Id	Assessment Type	Research Method	Development Contexts	Assessment Perspectives	Tool Chain Scenario
AS01	Analytical Study	Improving	Academy	P08	RAS++, RDL, FOMDA DSL and Brechó
AS02	Toy Example	Improving	Academy	P03, P06, P09	RAS++, MokupToME DSL, MAVEN and Eclipse Update-site
AS03	Toy Example	Improving	Academy	P03, P04, P06, P09	RAS++, MokupToME DSL and Eclipse IDE (Mylyn + ANT)
AS04	Analytical Study	Exploratory	Academy	P06	Representations for SPL tools in RAS and AMS
AS05	Toy Example	Improving	Academy	P03, P04, P06, P08, P09	RAS++, ReMoDD Artifacts and Eclipse IDE (Mylyn + ANT)
AS06	Case Study/MSR	Improving	Academy	P06	RAS++, MokupToME DSL and ReMoDD Artifacts

Table 1.5: Background discussing possible RAS++ assessments

Id	Reference	Title	Where?
PP01	BASSO (2015)	A Proposal for a Common Representation Language for MDE Artifacts and Settings	Doctoral Symposium - STAF
PP02	BASSO *et al.* (2015a)	A Summary of Challenges for "MDE as Service"	Workshop - WDES
PP03	BASSO *et al.* (2016b)	Criteria for Description of MDE Artifacts	Workshop - WDES
PP04	BASSO (2016)	Student Research Abstract: MDE as Service, Overview and Research Progress	Student Research Competition - SAC

Table 1.6: Mapping studies for RAS++ conception

Id	Title	Available at
M01	Characterizing the "MDE as Service" Research Agenda	Section 3.2
M02	Semantic Properties of Software Components	prisma.cos.ufrj.br/wct/ms02.pdf
M03	Intents from Asset Platforms and Their Properties	prisma.cos.ufrj.br/wct/ms03.pdf
M04	MDE Settings Intents and Their Properties	prisma.cos.ufrj.br/wct/ms04.pdf
M05	Diversity of MDE Toolboxes and Their Uncommon Properties	prisma.cos.ufrj.br/wct/ms05.pdf
M06	A Criteria for Representation of Technicalities from MDE Settings and Toolboxes	prisma.cos.ufrj.br/wct/ms06.pdf

Id	Review Type	Research Protocol	Sources
M01	Structured Mapping Study	Snowballing	Researchgate
M02	Ad-hoc Mapping Study	Key-wording	ACM, Researchgate
M03	Ad-hoc Mapping Study	Multi-vocal	Multiple voices
M04	Structured Mapping Study	Key-wording	Scopus, Researchgate
M05	Structured Mapping Study	Snowballing	Scopus, Researchgate
M06	Structured Mapping Study	Coding	Previous mappings

1.5 Thesis Organization

This thesis is organized in six chapters and one appendix with the following structure:

Chapter 1 - Contextualizes and motivates this research, pointing out to the problems and the proposed goals. The research methodology is also introduced.

Chapter 2 - Presents the theoretical foundation with general contributions in the area associated with MDE, tool chain, and assets.

Chapter 3 - Characterizes MDE as a Service and introduces our proposal built on three preliminary phases for tool chain and assets that guided the conception of RAS++.

Chapter 4 - Presents RAS++, our core contribution for representation of hybrid assets in coopetition scenarios for MDE as a Service.

Chapter 5 - Details four assessments composed by one case study on mining the ReMoDD repository and three analytical studies.

Chapter 6 - Outlines conclusions considering the goals and research questions.

Appendix A - Highlights the academic trajectory, summarizes some future works and closes this thesis with a personal position about the business opportunity in MDE as a Service, with perspectives to research and practice.

Chapter 2

Theoretical Foundation

> In the middle of difficulty lies opportunity.
>
> Albert Einstein

Model Driven Engineering (MDE) is an approach that considers models as first class citizens in software development (WHITTLE *et al.*, 2015). Typical artifacts in MDE (termed as **MDE Artifacts**) often include models, metamodels, model transformations, model managers, and domain-specific models. This chapter presents some fundamentals for MDE and Assets organized as follows. Section 2.1 presents some chaining mechanisms adopted for execution of model-based operations, Section 2.2 summarizes background information for assets, Section 2.3 discusses related works and Section 2.4 concludes this chapter.

2.1 Model-Driven Engineering

Approaches for MDE cope with complexity of software by using models. To SCHMIDT (2006), MDE has reduced the gap between solution and problem domains with abstract concepts that belong to the captured aspect of the object under study. Considering models as first class entities, which by the way are the main artifacts throughout the entire development process, a model can be used to support different engineering tasks. Many definitions for what model means are found in the literature. For example, Table 2.1 presents three definitions for models. Models, therefore, are abstractions that allow software engineers to focus on the relevant aspects of a problem domain while ignoring details that are irrelevant.

According to SOMMERVILLE (2010), MDE includes approaches for software development using models as abstractions for the systems solution domain (SELIC, 2006) and those using models as abstractions for processes (POLGÁR *et al.*, 2009). MDE is also associated with the use of automatic transformations from a model

specification to the solution domain. In the following, it is described the main concepts involved on the creation and execution of MDE approaches.

Table 2.1: Four definitions for model

Author	Definition	Year
(ROTHENBERG, 1989)	claims that for computer-related systems, a model is an abstraction from reality since it can not represent all the aspects of reality.	1989
(MILLER and MUKERJI, 2003)	*"A model of a system is a description or specification of that system and its environment for some certain purpose. A model is often presented as a combination of drawings and text."*	2003
(BÉZIVIN, 2005)	*"A model is a simplification of a system built with an intended goal in mind. The model should be able to answer questions in place of the actual system."*	2005
(BATORY *et al.*, 2013b)	*"We use ... (2) relational databases to express models"*	2013

2.1.1 Model-Driven Architecture

Behind MDE is the idea to raise the level of abstraction of the overall development process, to capture systems or processes as a collection of reusable models. Model-Driven Architecture (KLEPPE *et al.*, 2003) is an approach to realize MDE. It was proposed by the Object Management Group (OMG) and relies on their standard modeling and transformation languages to generate models, code and text from model specifications. In this approach, UML (BOOCH *et al.*, 2005) is the recommended language to represent the models, QVT (KLEPPE *et al.*, 2003) is the recommended language to write model-to-model transformations, and MOF Model to Text (OMG, 2008) is the language used to generate text/code.

Through inter-related views, this approach promotes the execution of tasks for modeling, refinement and transformations in interoperable design toolboxes. MDA toolboxes are connected through serializations in up to six versions for XML Metadata Interchange (XMI) (KLEPPE *et al.*, 2003).

Toolboxes can be complementary and competitors in these inter-related views. In this sense, Figure 2.1 shows views as different abstraction levels and their relationship for toolbox collaboration. Three levels of abstraction are suggested by MDA: Computation-Independent Model (CIM), Platform-Independent Model (PIM), and Platform-Specific Model (PSM).

The highest level of abstraction is implemented by toolboxes concerned in computation-independent model. Particular business domain models are described

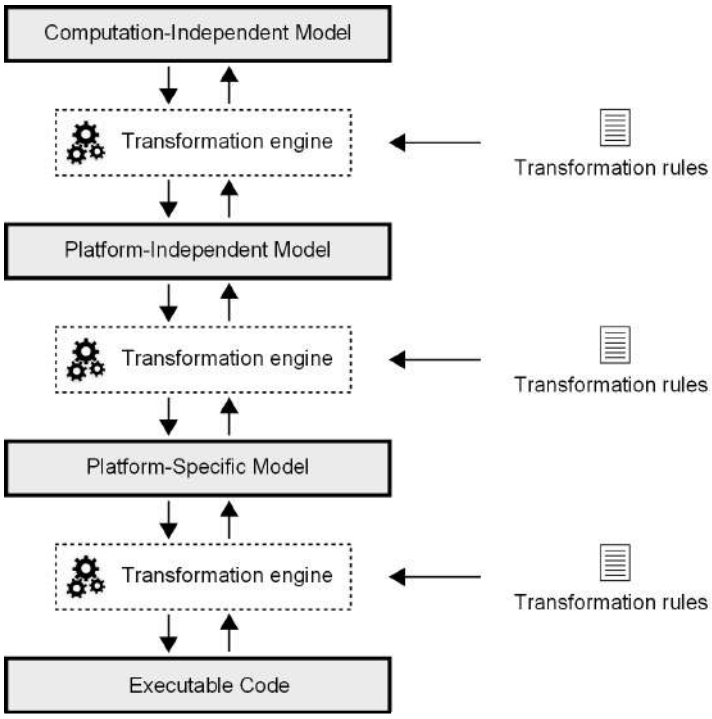


Figure 2.1: Abstraction levels in the MDA
(FRANTZ and CORCHUELO, 2012).

in this level of representation of software requirements. For this reason, specific information technology should not be represented in models in a CIM view. Software engineers shall use languages close to the abstractions of domain models (FERNANDES *et al.*, 2011), such as a Features Model (FM) (CZARNECKI, 2005).

In a second level of representation, toolboxes focusing in platform-independent models can refine a computation-independent model into application model instances (AZANZA *et al.*, 2010). At this level of abstraction, models are refined to describe the operations, structure, and behavior of a software system (BÉZIVIN *et al.*, 2004). Models shall remain without mappings to the underlining technology, and can be reused several times to generate different platform-specific models (KLEPPE *et al.*, 2003; KURTEV *et al.*, 2006).

Toolboxes supporting platform-specific models promote mappings from platform-independent models to implementation details (KELLY and TOLVANEN, 2008). This allows the generation of executable code, the final result from a transformation process in system engineering.

2.1.2 MDE Settings

According to HEBIG *et al.* (2013), the definition of MDE Settings characterizes MDE as part of many contexts in Software Engineering, from artifacts that represent a unique system model, to others that integrate multiple and heterogeneous

artifacts. For this reason, MDE is everywhere, promoting modeling in all phases of a Software Development Process (SDP) (KENT, 2002). For example, from a perspective of systems engineering (SOMMERVILLE, 2010), in the analysis phase, system models are usually general and with no information that can lead to a technical solution (SCHMIDT, 2006). In the following phases, system models are enriched with annotations containing details necessary to obtain the technical solution (ELKOUTBI *et al.*, 2006). These enriched system models can be used to enable automatic generation of source code (FRANCE and BIEMAN, 2001).

In order to illustrate how complex model-based approach can be, this section discusses a scenario of system engineering. In this scenario, model abstractions should represent enough details (i.e., annotations) allowing their use as input for model transformations, which by the way are performed with MDE Toolboxes classified as model transformation engines (JAKUMEIT *et al.*, 2014). Model transformation allows the generation of other models and, rich source-code using some toolboxes that can be characterized partially or entirely by elements shown in Figure 2.2. Finally, code is generated, refined by developers and compiled, resulting in working application pieces.

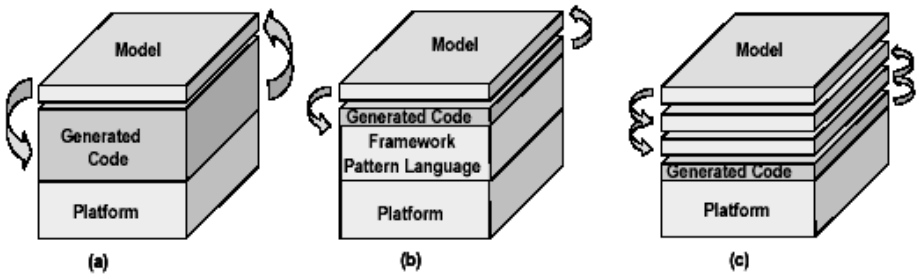


Figure 2.2: Different MDE implementations considering model to code
Credit: *GREENFIELD and SHORT (2003).*

To reach this point in a MDE-based process, many toolboxes and activities (manual, automatic or guided) are needed, adding extra complexity for the software development in comparison to a non-automatic approaches (BATORY *et al.*, 2013a). For example, only after rigorous refinement of models, software engineers can use transformations for code generation (OMG, 2008), allowing the transformation of abstractions represented in models to complete source code in many programming languages (KELLY and TOLVANEN, 2008).

Models are independent abstractions from code specificity (KENT, 2002). In order to meet the needs of a new target platform, a system model must be annotated with non-functional requirements (such as implementation details) that are different from one application to an other. Platform-specific details in models can

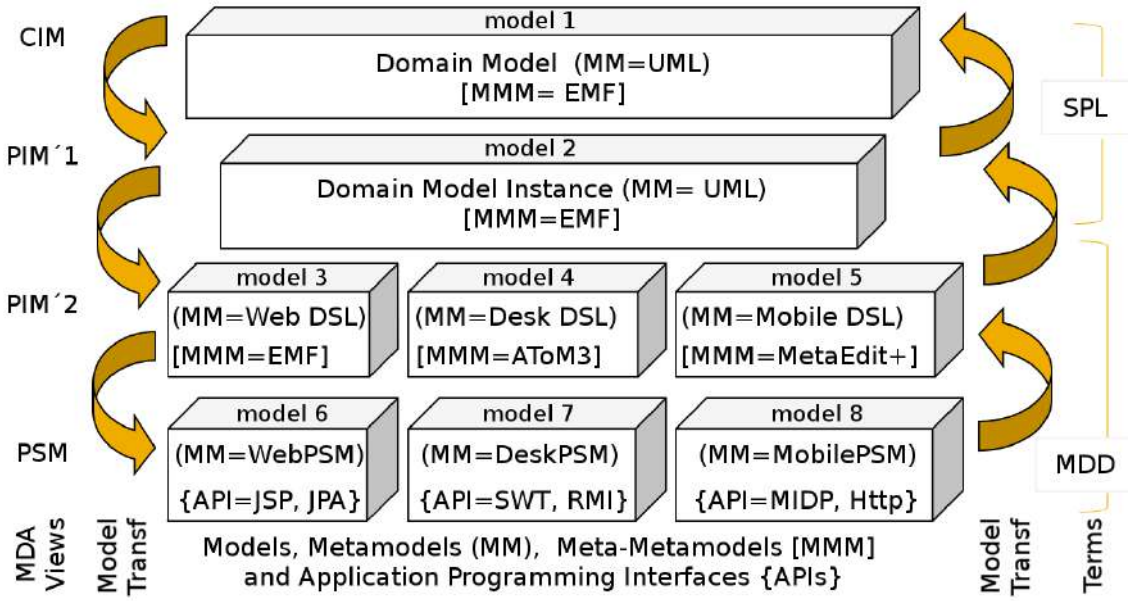


Figure 2.3: Model transformation lifecycle concepts

be generalized as `MDE Setting` (HEBIG, 2014), which determines platform's characteristics that demand specificity in code. MDE Settings are used by MDE Toolboxes for integration of model-based tasks including design, refinement and transformation. Moreover, MDE Settings are models too, and are typically constructed as DSLs (HEBIG, 2014).

2.1.3 Domain Specific Languages for System Engineering

According to BÉZIVIN (2005), the term "model" is dependent from the context in which it is applied. In system engineering, a model is the abstraction of system requirements. In this case, it is also called application model. Different types of systems may require different representation languages. This is the reason why the design of web information systems (SOUZA *et al.*, 2007) and embedded systems (BECKER *et al.*, 2002) implies the use of different Domain Specific Languages (DSLs) (VOELTER, 2009). Two main approaches are used to develop a DSL: Based on concepts for Domain-Specific Modeling Languages (DSML)(KELLY and TOLVANEN, 2008), also known as heavy-weight approach, and on UML Profiles (KLEPPE *et al.*, 2003), known as light-weight approach.

The literature of the area presents lots of DSLs for specific domains (FORWARD *et al.*, 2012). For example, some UML Profiles have been proposed to design web information systems (BLANKENHORN, 2004; KRAUS, 2007) and others to design embedded systems such as SPTP (OMG, 2002) and MARTE (OMG, 2013). These profiles are UML extensions used by software engineers to decorate model elements

with stereotypes and tagged values, providing application semantics in models.

In traditional software development approaches, software applications are manually programmed to run in specific target platforms. In order to create such applications independently of target platforms and using less manual effort from developers, some software companies are adopting code-generation techniques for MDA (BLOIS *et al.*, 2009; KLEPPE *et al.*, 2003; MONTEIRO *et al.*, 2014a). For example, in a MDA-based process a model, in general, represents elements using UML (BOOCH *et al.*, 2005) or other languages whose metamodel is based on the Meta Object Facilities (MOF) (OMG, 2008). MDA' views (CIM, PIM and PSM) are semantically helpful to manage tasks of model refinement in levels of details, such as those illustrated in Figure 2.1, but in a concrete instance for tool chain shown in Figure 2.3. In this sense, a concrete tool chain should include a modern MDE Toolbox that supports Software Product Lines (SPL) (KANG *et al.*, 1990, 1998) in a CIM view. Many toolboxes support for software product lines (THÜM *et al.*, 2014), allowing the generation of software products (models and code) based on variability and commonality found in an application domain (FERNANDES *et al.*, 2011). This example makes use of a "pattern language" (BATORY *et al.*, 2008) to generate domain independent products (e.g., embedded systems or web information systems) in conformity with UML.

Batory claimed that the implementation of SPL and Model-Driven Development (MDD), or MDSPL, requires the use of heterogeneous model transformations because more than one DSL is used in a MDSPL lifecycle (BATORY *et al.*, 2008). For example, the transformation of a domain model from a CIM view to another model in a PIM view (the `Domain Model Instance`) does not require the use of an heterogeneous transformation, while the other requires. Since "model 2" of Figure 2.3 is still in conformity with UML, a transformation from "model 1" to "model 2" is defined as an endogenous transformation (i.e., that makes use of the same metamodel). In the PIM'2 view, the Domain Model Instance is transformed to: a) model 3, conforms to the Web DSL; b) model 4, conforms to Desktop DSL; and c) model 5, conforms to Mobile DSL. This management of heterogeneity can, therefore, also be mapped to constructors of DSLs. Thus, approaches for MDE Settings do not follow strictly the nomenclature defined by MDA neither are obligated to adopt UML as unique design language.

2.1.4 Lifecycle for Transformations

Since 2000, development effort has been directed to create DSLs and different model transformation (MT) engines (JAKUMEIT *et al.*, 2014). In 2003, Czarnecki et al. made a classification of transformation proposals (CZARNECKI and HELSEN,

2003), dividing approaches in two categories: Model-to-Code (M2C) and Model-to-Model (M2M). In M2C transformations, the goal is to generate code directly from models or templates. In M2M transformations, the objective is to transform the input model into another model. Authors claim that several model transformations are necessary to generate system code, and such transformations are managed through tool support (BÉZIVIN *et al.*, 2004).

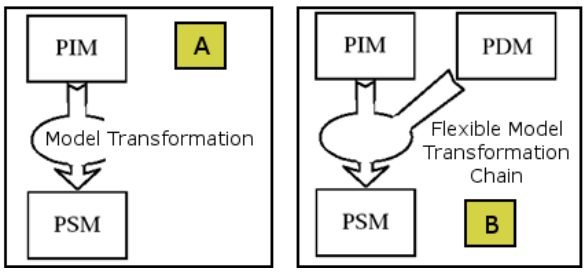


Figure 2.4: Mapping and Transformation of a Model in a PIM View to a PSM View

Instead of using only one transformation directly from a model to code, the literature suggests the use of several transformations (MENS and GORP, 2006). Figure 2.2 shows three possible implementations to transform a model to code and also to reverse code to model (known as round-trip engineering (HAILPERN and TARR, 2006)). Early implementations have followed the structure shown in Figure 2.2 (a). In this structure, a model is converted to code using a tool that operates for a specific platform, and reverse engineering converts the code to model. In Figure 2.2 (b), concepts for frameworks and pattern languages are added, allowing code generation considering core assets in a target context. FERNANDES *et al.* (2011) exemplified a MDE implementation through software product lines, which shows the use of a pattern language to generate products independent from domains (e.g., embedded systems or web information systems). Batory claimed that the implementation of Figure 2.2 (c) requires the use of heterogeneous model transformations because more than one DSL is used in a MDE lifecycle (BATORY *et al.*, 2008). Figure 2.2 (c) shows a possible implementation of modern MDE tools, which enables a tool chain approach independent from frameworks and languages.

In order to provide semantics for model transformations in multi-levels, some authors recommend the use of a fourth view called Platform Description/Domain Model (PDM) (BASSO *et al.*, 2006; WILLINK, 2003), as illustrated in Figure 2.4 (B). A PDM contains mainly characteristics of available platforms (e.g., operation systems, application programming interfaces, libraries, frameworks, codification patterns, and others). It also can present any semantic that can be associated with a model transformation. To transform a PIM into a PSM, for example, it is necessary to identify platforms that can make this refinement. In 2006, we presented an ap-

proach to combine characteristics from PDM with model transformations (BASSO *et al.*, 2006), adding flexibility in model transformation processes through Feature Model (FM) (CZARNECKI and HELSEN, 2003; KANG *et al.*, 1990).

A possible implementation of a complete tool chain approach is illustrated in Figure 2.3. Several DSLs proposed in the literature are based on software product lines reuse mechanisms, allowing the specification of models in a CIM view and transformation to a PIM view. Following related works can be mentioned: (BENAVIDES *et al.*, 2010; FERNANDES *et al.*, 2011; KANG *et al.*, 1998; VÖLTER and VISSER, 2011).

In addition to software product line approaches, a multi-level development approach also considers model refinement with heterogeneous metamodels, as illustrated in Figure 2.3 by two levels of PIMs. In level PIM 2, many DSLs allow generating GUI models for information systems. Examples are DSLs proposed by (KAVALDJIAN, 2007; KRAUS, 2007; NUNES and SCHWABE, 2006; VANDERDONCKT, 2005). This way, multi-level modeling needs adaptations on MDE Settings (e.g., connected model transformation and tools in chains) for a more heterogeneous scenario (e.g., to combine components from textual DSLs such as ATL (BÉZIVIN, 2005) with compiled Java components).

2.1.5 Tool Chain Approaches

Some works implement both SPL and MDE. In (BATORY *et al.*, 2008), authors introduced an approach called "model deltas" as a possible implementation for Model-Driven SPL (MDSPL). MDSPL approaches are based on the megamodel concept (MARIE FAVRE and NGUYEN, 2004), which requires modeling of everything (variability in domains, in transformations and any other resource used in the transformation process) in a unique big model. A megamodel simplifies the representation and management of variability (MARIE FAVRE and NGUYEN, 2004; VIGNAGA *et al.*, 2013), but its use is limited in inter-organizational contexts, where model transformation components are used in different domains and adapted for different needs in target companies. Instead of using a megamodel, we have proposed adaptations in model transformation chains, which have presented positive results in practice considering reuse in inter-organizational contexts (BASSO *et al.*, 2013a, 2017a). A negative aspect of this proposal its higher complexity than MDSPL, requiring adaptations that consider features from cross-domains (e.g., reusable components between model transformations for web information systems and wireless sensor networks).

Batory discussed important concepts for hybrid representations of model-based operations (BATORY *et al.*, 2008). In regard to model transformations as applied

in MDSPL, two main classifications exist: Endogenous versus Heterogeneous. Endogenous transformations are used in software product lines and make use of the same metamodel (e.g., the input "domain model" and the output "domain model instance"). Heterogeneous model transformations are more complex and need verification in a chain of model-based operations. This is illustrated in the bottom-part of Figure 2.3, where some heterogeneous transformations are executed after the generation of a domain model instance. Transformations are also classified according to its complexity for MDSPL: more complex transformations make use of "ad-hoc algorithms" and less complex ones make use of "built-in algorithms" and "generic algorithms". Heterogeneous transformations make use of many "ad-hoc algorithms", that may be in conformance with more than one metamodel, may simply be programmed in Java.

Representations for MDE Settings enable the execution of automated design approaches (BATORY *et al.*, 2013b; HEBIG, 2014) independently from frameworks and languages, allowing the management of model transformations as components in an execution sequence. For example, Figure 2.3 shows arrows indicating flows of execution that chain model-to-model and model-to-code transformations. A concrete example is introduced by LEVENDOVSZKY *et al.* (2005), which presented the Visual Modeling and Transformation System (VTMS) as a solution for chaining several transformations that generate refined models conforming to different views. His work is limited to an unique metamodel (i.e., views are models represented with UML), thus restricted to MDA Settings (UML, XMI and OCL standards).

For managing of orchestration of heterogeneous meta-metamodels, other contributions are dedicated to check consistency. CAPLAT and SOURROUILLE (2005) highlighted that the real issue is to accomplish mappings from one language to another (i.e., to link and validate model, metamodels and transformations). First approaches for MDE Settings presented limitations to implement a more heterogeneous tool chain. Modern tools supporting reuse mechanisms have been adopting validation techniques, such as (YIE *et al.*, 2012), using different representations/settings. Thus, these techniques remit to specific settings that need to be analyzed in this work.

It is important to mention that tool chain is not an issue observed only in MDE context. In this sense, ZAKHEIM (2017) presents the following issues associated with tool chain in general contexts:

> *An unintegrated software development and delivery toolchain creates bottlenecks, drains productivity, impedes collaboration and inhibits project visibility. Integration of the toolchain may seem like a straight forward undertaking, but it is much more difficult than it appears initially. While most endpoint tools have APIs, they are insufficient for integrating to*

> *other tools because they: were not designed for this kind of integration, are often poorly documented and often change when a new version is released. Some of the most challenging aspects of integrating these tools include understanding how practitioners use these tools and resolving the conflicts among the tools that prevent this flow of work. Managing the performance impact, testing and going maintenance of an integrated toolchain becomes exponentially more difficult the more endpoints are added.* (ZAKHEIM, 2017)

ZAKHEIM (2017) also states that business issues rather than technical are currently affecting the success of tool chain approaches. In this sense, next, we introduce the business concept for coopetition, presenting assets as modern and complementary elements in support for implementations of tool chain in MDE.

2.2 Assets

The literature is rich in toolboxes that support tool chain instantiation, modeling, refinement, and transformation tools. However, putting these toolboxes to work together is always a difficult experience (MOHAGHEGHI *et al.*, 2013). As the leader of the company Tasktop, which focuses on the integration of software development tools, ZAKHEIM (2017) concludes that the lack of semantics of tools is a problem hampering their work.

A common concept for introducing the semantics for artifacts is the asset. An asset is anything that provides reuse and value through reference (links), cataloged with standardized taxonomies, described by a set of properties and owning zero or more specifications about artifacts. The Gartner Group, an important organization that evaluates industrial practices in Software Engineering, has conducted a survey of companies stating that the assets are considered essential to their business (GARTNER, 2013). Assets are, therefore, relevant to the current reuse scenario.

Through ad-hoc reviews, we find in the literature of the area the subject "asset" discussed in four different perspectives. Thus, this section presents the following concepts related to assets: asset specifications, coopetition, repositories and software ecosystems.

2.2.1 Asset Specifications

A possible solution for publishing and downloading artifact content in a repository is by asset specifications (OMG, 2005). According to the Asset Management Specification (AMS) (AMS, 2014), a resource is anything that provides reuse and value through a reference, cataloged with taxonomies, described by a set of properties,

and having zero or more data for artifacts (AMS, 2014). Assets have been adopted to describe software components (HONG-MIN *et al.*, 2009), application and domain models (PARK *et al.*, 2007) and, more recently, tools shared in the cloud in the context of MDE (ELAASAR and NEAL, 2013). DOS SANTOS and WERNER (2010) claims that assets are important to play a key role between reusable artifacts (asset consumers) and service providers/repositories (asset providers).

The OMG (Object Management Group) promotes a standard for asset representation called Reusable Asset Specification (RAS) (OMG, 2005). This standard allows classifying, cataloging, and instructing the reuse of software artifacts in arbitrary reuse repositories, such as Rational Asset Manager (MADSEN, 2008) and OpenCom (HONG-MIN *et al.*, 2010). The data provided in reusable assets allow one to specify, store, retrieve various artifacts used in the software engineering process. The objective of the asset producer is therefore to detail data associated with artifacts, such as instructional information on how to adapt and deploy a set of artifacts in production environments.

2.2.2 Coopetition

Future implementations in MDE as a Service can benefit from promoting competition promoted through web services (OSLC, 2017b). In this section, we present some initiatives that are gaining attention in this matter.

Cooperation and competition appear to be conflicting terms, but practice proves otherwise. The case of Amazon.com is a good example of cooperation and competition (RITALA *et al.*, 2014). Since this company opened its own online store for third-party sellers (competitors), new business transactions increased 20 percent of its North American sales between 2002 and 2010 and other 35 percent of sales in 2013. RITALA *et al.* (2014) claims that:

> *"Coopetition (collaboration between competing firms) is a phenomenon that has recently captured a great deal of attention due to its increasing relevance to business practice."*

BOUNCKEN *et al.* (2015) define coopetition as:

> *"Coopetition describes an interorganizational relationship that combines "cooperation" and "competition".*

RITALA *et al.* (2014) report three distinct coopetition-based business models implemented at Amazon.com. The second model "(2) Services and Web Services" is what interests us since it is feasible for implementation in MDE as a Service. Services and Web Services remit to the definition of a common representation for

products commercialized by Amazon. These products are represented as asset in the Amazons' cloud, which are connected with external competitors through web service implementations (RITALA *et al.*, 2014).

2.2.3 MDE Artifact Repositories

Aware of this trend for coopetition (BOUNCKEN *et al.*, 2015; RITALA *et al.*, 2014), some authors suggest the use of an underlying infrastructure that promotes the reuse of MDE artifacts through assets (COMBEMALE *et al.*, 2015a; MUSSBACHER *et al.*, 2014). In this direction, some options for global asset repositories concerning MDE Artifacts have been in operation since 2006 (FRANCE *et al.*, 2006). ReMoDD is a repository for MDE that can be used to solve the problem of reuse and sharing of MDE Artifacts for free (FRANCE *et al.*, 2007). An initiative called SEMAT focuses on reusable methods (JACOBSON *et al.*, 2012) represented with an OMG's standard called Essence (OMG, 2015). GEMOC (COMBEMALE *et al.*, 2014) is another option. Finally, recent underlying infrastructures represent technical aspects for MDE Artifacts in repositories that support collaboration (MENGERINK *et al.*, 2016; ROCCO *et al.*, 2015).

2.2.4 Software Ecosystems

Assets have also been suggested as essential in Software Ecosystems (SECO) AXELSSON *et al.* (2014); SANTOS and WERNER (2012). At SECO platforms, assets are typically analyzed from a three-dimensional perspective (technical, business, and social). Although limited to use in the context of MDE artifacts, some mapping studies for SECO are also considering elements for decision making (AXELSSON *et al.*, 2014; MANIKAS, 2016; MANIKAS and HANSEN, 2013).

Some platforms for Software Ecosystems propose support for competition through negotiation (AXELSSON *et al.*, 2014; BADAMPUDI *et al.*, 2016; DOS SANTOS *et al.*, 2013; SANTOS *et al.*, 2016). These platforms enable interactions between asset providers and consumers. SANTOS *et al.* (2016) present a distribution channel as a possible solution for trading assets in global and corporate component markets, which is also valid in the context of the MDE. According to BADAMPUDI *et al.* (2016), assets are critical to decision making of software components and comprise the following reuse scenarios: 1) in-house, i.e., considering intra-organizational contexts, and; 2) inter-organizational scenarios, i.e., including Components-of-the-Shelf (COTS), Open Source Software (OSS), or outsourcing.

2.3 Related Works

In this section, we highlight the main related works in scenarios of MDE as a Service. In order to capture the basic terminology concerned with services, assets, pivot (or pivotal) representations, we performed an ad-hoc technical literature review using the researchgate.net engine as data source. Besides, we selected some contributions for tool chain discussed previously, organizing this section is organized as follows. Section 2.3.1 discusses proposals for model transformation chain and reuse mechanisms. Section 2.3.2 reports on recent advances from proposals for representation of tool chain in MDE. Section 2.3.3 presents some approaches for services and assets. Section 2.3.4 highlights main extensions for asset specifications and Section 2.3.5 presents studies for assets used in software ecosystems. Section 2.3.6 establishes the relation of our work with reference architectures and Section 2.3.7 exposes MDE repositories as important players for assets in coopetition scenarios for MDE as a Service. Finally, Section 2.3.8 encloses with related works for pivotal representation languages in MDE.

2.3.1 Reuse Mechanisms in Tool Chains

Reuse mechanisms for tool chain are related with the toolbox built on the FOMDA DSL (BASSO *et al.*, 2013a, 2014c,e), a representation language that we have used to build foundations for MDE as a Service through characterization studies (BASSO *et al.*, 2014a, 2015b, 2016d, 2017a). Thus, below we describe related works to this topic.

Aiming to generate Model Transformation Chains (MTCs) used by MDE tools for execution of model transformations, some proposals allow designing transformation compositions in high-level of abstraction to then generate specific scripts. VANHOOFF *et al.* (2006) proposed a MTC modeling language using target platforms as part of architectural designs. Target platform and transformations are composed using UML activity diagrams in a similar solution as we did in previous contributions, using a UML Profile (BASSO *et al.*, 2006). These proposals were complemented by WAGELAAR (2006) who proposed a composition of black-box model transformations and by ETIEN *et al.* (2012) who suggested the use of heterogeneous model transformations developed with different metamodels. Although Etien's and Vanhooff's proposals are interesting, they do not support transformation chain executions considering target platform variants, meaning that chains must be manually changed when a different set of transformations is necessary.

Another recent contribution supporting execution of MTCs is presented by VARA *et al.* (2014), which considers the trace promoted by model transformations but does not consider variability in these transformations. In opposite,

FODMA DSL does not consider traceability, but its tool chain representations consider the existence of two types of runtime-based adaptations: the first one interprets a model specification for tool chain and the second one integrates variants directly into scripts.

Some recent proposals have applied similar reuse techniques through generative approaches. ARANEGA *et al.* (2012a) uses the Feature Model to assemble transformation components (e.g., white-box model transformations, fragmented algorithms or transformation rules, diverse configuration files, etc) using Software Product Line (SPL) tools. However, despite techniques for SPL are applicable in diverse component types, adaptations in tool chains must also consider valid compositions among artifacts. While ARANEGA *et al.* (2012a) and ETIEN *et al.* (2012) seem to move in this direction, so far, only FOMDA DSL and Bentõ DSL (CUADRADO *et al.*, 2014) consider the use of Feature Model with verified components.

The toolbox built on the FOMDA DSL is the only one that validates the constructors from three different techniques (SPL, MTC, and Component-Based Development - CBD) at design and runtime. Thus, compositions are checked during the design of a domain model for MTC, while in (ARANEGA *et al.*, 2012a; CUADRADO *et al.*, 2014; ETIEN *et al.*, 2012; YIE *et al.*, 2012) they are checked only after the generation of a concrete model transformation chain/script. In comparison to (ARANEGA *et al.*, 2012a), FOMDA DSL toolbox saves time in detecting inconsistencies in the design of components. Considering consistency in transformation composition, GUY *et al.* (2012) and YIE *et al.* (2012) suggested that valid compositions must be ensured with parameter types. For example, IO parameters are checked during the design of a model transformation chain, validating rules such as metamodel and data types (e.g., EMF-based metamodels compatibility), transformation languages (e.g., ATL and QVT transformations), etc. Moreover, it is also important to consider that model transformation components can evolve. In this regard, LOPEZ-HERREJON *et al.* (2010) presented a proposal to automatically control co-evolution of models, metamodels, and transformations from a tool chain. Although co-evolution is an important technique related to adaptations in MDE Artifacts, it is not managed in FOMDA DSL toolbox neither in any toolbox developed in support for RAS++.

FOMDA DSL is complementary to RAS++. It contains an integrated toolbox for construction of tool chains, consisting of an approach for integration, likewise previously discussed contributions. On the other hand, RAS++ is for preliminary phases to integration, using a rich set of structures to describe assets. Thus, these two types of DSLs are complementary.

2.3.2 Tool Chain in Process Engineering

This section describes related work in tool chain supporting process engineering.

LAHTINEN *et al.* (2006) propose to guide the refinement of UML models by assisting the creation of object diagrams based on a class diagram used as input. They used the MADE platform (HAMMOUDA, 2005), that is based on a UML profile to specify MDD tasks. RAS++ presents similar resources (BASSO *et al.*, 2014b).

Other integration approaches are for process specifications, or Process Modeling Languages (PMLs). POLGÁR *et al.* (2009) proposed a framework to chain transformations by chaining tools through a reuse process defined with SPEM, a similar approach for tool chain such as proposed by MACIEL *et al.* (2013). OLIVEIRA *et al.* (2011) assist some tasks to instantiate object oriented frameworks using BPMN, incremented recently for inclusion of workflow elements (LUCAS *et al.*, 2017). Besides, many other works that use model transformation engines to execute transformation processes are discussed in (BASSO *et al.*, 2013a).

Approaches for integration of MDE Artifacts and Settings in process models are used in a late phase of tool chain unlike RAS++, which is useful in preliminary phases. Our contribution focuses on phases Specification, Acquisition and Transformation, which are little understood in the literature and practice on tool chain (ZAKHEIM, 2017). MDE assets are agnostics to the software development process model, process modeling language, MDE Toolbox and integrated development environment. Existing works consider all these elements integrated (e.g., ETIEN *et al.* (2012); MACIEL *et al.* (2013); YIE *et al.* (2012)), whereas in a coopetition scenario as implementation for MDE as a Service these elements need to be managed independently. In order to promote reuse of assets for different contexts (i.e., considering different formats/solutions used to integrate MDE tasks), we first extracted from the literature requirements for a common representation language. Such a language is imperative for execution of preliminary phases and represents the main contribution of this thesis.

2.3.3 Anything as Service

Although little related, there is a soft connection between our work and some approaches for anything related to services. Service-oriented product lines (CASTELLUCCIA and BOFFOLI, 2014) are approaches for management of variability in artifacts built on concepts of Service-Oriented Architecture (SOA). These contributions have little relation with RAS++, being complementary. There is also a difference in terms of general goals: MDE as a Service aims at providing software engineering services for customization of MDE Artifacts, while approaches for service-oriented

product lines focus on reuse of service components shared on the web.

Typically, works related to RAS exemplify scenarios where assets promote reuse through a well formed documentation. It is the case of ELGEDAWY (2009), who proposed a RAS extension to describe SOA components, and PARK *et al.* (2007) that detail component interfaces. The goal of these works is to make reusable assets for web services. Properties introduced in RAS++ did not consider other elements than those necessary for MDE Artifacts and Settings. This means that for representation of SOA components, asset profiles described by ELGEDAWY (2009); PARK *et al.* (2007) must be included using one of two options in RAS++: UML Profiles or RAS Profiles.

This is not the case for federation of resources. In this sense, a research in evidence is "MDE Resources as Service" (BASCIANI *et al.*, 2014b), focused on implementation technologies for Software-as-Service (SaaS) inside MDE repositories. SaaS allows to publish and download resources to/from repositories through modern APIs for services, such as REST API (ROCCO *et al.*, 2015). Investigated repositories are more general and, therefore, RAS++ considers other elements for artifact federation (local path, URLs, MAVEN proxies, OSLC proxies) instead of a unique SaaS technology. Thus, in representational terms, RAS++ is more representative than the proposal of BASCIANI *et al.* (2014b).

2.3.4 Extensions for Asset Specification Languages

RAS++ is built on service concepts in asset-level, as required by the standards RAS (OMG, 2005) and AMS (AMS, 2014). The RAS standard is older and defines a format for web services, allowing access to component-based repositories. The AMS standard defines a format for accessing clouds, including modern definitions such as REST API and Resource Description Framework (RDF). RAS++ is built on RAS and AMS standards, but so far none of the prototypes that we have developed support the execution of services such as query mechanism. For this reason, prototypes for RAS++ are in level of DSL and not in implementation-level of web services, that run with RAS or AMS protocols. For such purpose, software engineers should transform RAS++ representations to RAS or AMS specifications.

Concerning representativeness from asset specification languages, we can mention some extensions for RAS that represent technical and descriptive information: SOA components (ELGEDAWY, 2009; PARK *et al.*, 2007), feedback from users (HADJI *et al.*, 2008), component software license (HONG-MIN *et al.*, 2009) and standard taxonomies. Other proposals (BIEHL *et al.*, 2014; ZHANG and MOLLER-PEDERSEN, 2014) are more similar to integration languages whose specifications can be transformed to AMS (AMS, 2014), which is a standard behind the Open

Services for Lifecycle Collaboration (OSLC) (OSLC, 2017a). These extensions can also be applied to summarize information of some technical data for MDE. However, they are used by software components or tools shared on the cloud, thus not properly designed to represent technical information associated with MDE Artifacts and Settings.

Anyway, some of these contributions can also be used in preliminary phases, but closer to integration. Considering chained execution of tasks integrated in target environments like Eclipse IDE, BIEHL *et al.* (2014) proposed TIL, a domain specific language to specify integration among tools. Besides, integration adapters based on TIL are used to generate OSLC specifications used to integrate diverse assets found in software development scenarios. ZHANG and MOLLER-PEDERSEN (2014) proposed to detail artifacts and roles in the context of tool integration, also applying mappings to OSLC specifications. Clearly, the authors are also moving towards detailing assets in preliminary phases. However, our contribution is singular because we have considered specificity from MDE tool chains and also quality of the descriptive information, which is superficially discussed by aforementioned works.

2.3.5 Software Ecosystems

Other contributions related to assets propose better descriptions concerning business issues in perspectives for software ecosystem (SECO) (AXELSSON *et al.*, 2014; BADAMPUDI *et al.*, 2016; DOS SANTOS *et al.*, 2013; SANTOS *et al.*, 2016). These descriptions are possible only on asset platforms, therefore, not through DSL or a common representation. In this regard, BADAMPUDI *et al.* (2016) propose reused dimensions for ecosystems, including the assets as an important player for decision making in a reuse approach called GRADE. RAS++ allows the representation of the information suggested in GRADE. However, our assessments are focused in technical information, thus making a direct transition from the GRADE phases to the decision making: Assets and Environment.

SANTOS *et al.* (2016) present a distribution channel as a possible solution for the negotiation of assets in global and corporate component markets, which is valid in the context of MDE as a Service. With this intention, the authors have promoted the use of three perspectives in the SECO platforms (LIMA *et al.*, 2016) representing assets mapped to describe: socio-technical and business opportunities. These contributions influenced the criteria established for qualified descriptive data at the Acquisition phase. To our best knowledge, in terms of DSL, only RAS++ match these recommendations relevant for decision making from the point of view of technical, business and social actors on the representation of assets for MDE Artifacts and Settings.

2.3.6 Reference Architectures

Some related works (MARTINEZ-FERNANDEZ *et al.*, 2015; NAKAGAWA *et al.*, 2015) suggest reference architectures as a common representation for instantiation of MDE Artifacts such as system models, system code, interfaces to Web Services, and other applications that can be built on a unique conceptual model. For example, NAKAGAWA *et al.* (2015) suggested reference architectures as a way to promote the connection of concepts to representations for systems-of-systems (NETO *et al.*, 2014), which is ultimately used for web services definitions that connects heterogeneous systems.

Reference architectures can defined to instruct the integration of diverse types of assets in MDE as a Service as well, complementing asset specifications. Reference architectures supporting MDE as a Service could be structured as RAS++ assets as well. Moreover, since heterogeneous services are the key to interoperate asset information in MDE as a Service and are part of reference architectures, we believe that a more general reference architecture could guide the implementation of specific web services that connects heterogeneous sources in a coopetition scenario with the common representation in RAS++.

Reference architectures, if we had found some in the literature, would be very useful and spared effort to achieve common RAS++ properties. The process for achieving these properties in the reference architectures is the same adopted in this thesis: the execution of coding protocols for clustering. Therefore, the use of reference architectures in support for MDE as a Service is an interesting open topic for investigation.

2.3.7 Infrastructures Sharing MDE Artifacts

In 2006, Boehm argued that to compete, adapt, and survive, software development companies will depend on the ability to integrate some systems into global reuse scenarios made of Systems of Systems (SoS) (BOEHM, 2006). In 2014, Fuggetta highlighted a trend for cooperative systems that assist software engineering tasks in diverse software development phases (FUGGETTA and NITTO, 2014), thus in integrated processes. This trend is of interest for the MDE context (HEBIG and BENDRAOU, 2014), and in special of interest of MDE as a Service. Industry is requesting some facilities to introduce MDE-based solutions in practice (WHITTLE *et al.*, 2015), such as more compatible systems (LIEBEL *et al.*, 2014) and qualified information (MUSSBACHER *et al.*, 2014). Therefore, besides SECO (DOS SANTOS *et al.*, 2013) and reuse repositories (COMBEMALE *et al.*, 2014; FRANCE *et al.*, 2007; JACOBSON *et al.*, 2012; ROCCO *et al.*, 2015), contributions for interoperation of SoS (MOTTA *et al.*, 2017) are important for execution of preliminary

phases for integration.

Some efforts to achieve interpolations between SoS and SECOs presented similarities and differences (JERONIMO JR. and WERNER, 2015): the former includes business prospects, while the latter is quite technical. From a personal opinion, SoS is more at the OSLC (OSLC, 2017b) implementation level, including services and interfaces, rather than on a descriptive level of SECO. Contributions to SoS were also discussed in the context of MDE (NETO *et al.*, 2014). For example, design and refinement tools are MDE Artifacts considered as systems (FRANCE *et al.*, 2006), as well as some DSLs with an associated plugin or tool support (COMBEMALE *et al.*, 2014). The integration of these systems occurs through the MDE Settings (HEBIG *et al.*, 2013), which are executable compositions of model transformations (HEBIG, 2014). Last but not least, model transformations are also systems (BATORY *et al.*, 2008).

These approaches are discussing the same problem with different representational focus, some considering more important technical aspects and others stating the importance of descriptive data. In this way, representations found in SoS-based repositories are related works (CORREIA *et al.*, 2016; NAKAGAWA *et al.*, 2015). Its focus is on a reuse scope also implemented by the MDE repositories (COMBEMALE *et al.*, 2014; FRANCE *et al.*, 2007; JACOBSON *et al.*, 2012; ROCCO *et al.*, 2015) and by the SECO platforms (DOS SANTOS *et al.*, 2013), but addressing more technical issues than others related to decision making. Although limited to a few MDE configurations, compared to the above-mentioned repositories, we found that MDEForge (BASCIANI *et al.*, 2014b) provides representations that best match some of the tool chain components exemplified in this thesis. However, because of the nature of shared MDE artifacts (some share DSLs, other model transformations, other systems, etc.), adopting a unique repository in MDE as a Service is unlikely. Thus, each infrastructure presents opportunities for reuse and should be considered in competitive scenarios in MDE as a Service.

Our long-term research effort is to integrate them all through a common representation, starting with the specificity of MDE. For this reason, we consider the RAS ++ metadata as a super-set of all those infrastructures that support global reuse. RAS++ could map representations found in these infrastructures, but first we need to perform an appropriate assessment for each of them, as we did in the ReMoDD repository (FRANCE *et al.*, 2007).

2.3.8 Pivotal Representations in MDE

Considering the four reuse phases introduced by KRUEGER (1992), bellow we include comparison with "pivotal" representation languages for MDE Artifacts and

Settings.

To the best of our knowledge, RAS ++ is the unique DSL integrating concepts from assets and tool chains for pivoting repositories and development environments (BASSO *et al.*, 2017b). The state of the art in pivot languages for MDE is focused on: 1) Architectural Description Languages (ADL) supporting model transformations (SYRIANI *et al.*, 2015), which bridge known languages such as Atlas Transformation Language (ATL) (BÉZIVIN, 2005) and Epsilon Transformation Language (ETL) (KOLOVOS *et al.*, 208) in a common representation; 2) Reuse mechanisms built on Component-Based Development (CBD) concepts (HERMER-SCHMIDT *et al.*, 2013); 3) Reuse of model transformations built on common concepts (CUADRADO *et al.*, 2014); and 4) common constructors for ADLs in the level of DSLs (DI RUSCIO *et al.*, 2012).

Existing representations for assets or tool chains are limited to play a pivotal role required in the MDE as a Service: 1) due to its specificity for concepts and reuse mechanisms specific to the MDE Settings (BASCIANI *et al.*, 2014b; BASSO *et al.*, 2014e; CUADRADO *et al.*, 2014; HEBIG *et al.*, 2012; JOUAULT *et al.*, 2010; MAS-CARENHAS *et al.*, 2013; POLGÁR *et al.*, 2009; YIE *et al.*, 2012); 2) due to the specificity of reuse mechanisms of MDE Artifacts when associated with structural features from a repository (BIEHL *et al.*, 2014; CRIADO *et al.*, 2015; ELAASAR and NEAL, 2013; GARCÉS *et al.*, 2014; LÚCIO *et al.*, 2014; VIGNAGA *et al.*, 2013; ZHANG and MOLLER-PEDERSEN, 2014), and; 3) due to the fact that these representations do not match because they are in different formats, including database structures in repositories (BASCIANI *et al.*, 2014a; FRANCE *et al.*, 2007), XML (MONTEIRO *et al.*, 2014a), code (VÖELTER and GROHER, 2007), textual (OLIVEIRA *et al.*, 2011) or graphical DSLs (VIGNAGA *et al.*, 2013).

Thus, our contribution is singular since RAS++ groups a greater number of structural features needed in future implementations for MDE as a Service.

2.4 Final Remarks

This chapter summarized essential concepts for the follow-up of this thesis. The next chapter is a complement, discussing these concepts as important for implementation of an emerging scenario for Software Engineering called MDE as a Service (MDEaaS), i.e., a business opportunity characterized by requests from application-independent contexts for customizations in MDE Artifacts. In response for the increasing interest of some professionals in this opportunity and due to the absence of studies establishing reuse foundations for this scenario, a summary of our characterizations for MDE as a Service is presented with the intent to introduce our main contribution: RAS++.

Chapter 3

MDE as a Service

> The only real valuable thing is intuition.
>
> Albert Einstein

This chapter presents our contributions to the foundation of MDE as a Service (MDEaaS) through a summary of four longitudinal studies and six structured reviews of type mapping study, organized in sections as follows: Section 3.1 introduces the contexts of the longitudinal studies, presenting methodological concerns for implementations of MDE as a Service. Section 3.2 highlights current research effort considering diversity in MDE Artifacts and Settings extracted from a mapping study with implementation concerns. Complementary mappings are summarized in Section 3.4 exposing our concerns for cooperation and competition in the context of MDE as a Service. Section 3.4 presents our concerns for the conception of a common representation language for assets, and Section 3.4.4 encloses this chapter with final remarks for conception of RAS++.

3.1 Methodological Concerns

This section discusses experiences from previous implementations of MDE as a Service. As illustrated in Figure 3.1, resources developed for MDE (e.g., model transformations, DSLs and tools) are applied in different contexts. MDE as a Service was implemented through the company *Adapit*, founded in 2007 and supported for three years by a business incubator, hosted in one of the biggest scientific and technological parks in Brazil. Through Adapit, MDE resources were introduced in five software projects, three of them developed by teams from Adapit (context of box 1 in the figure) and two out of them by teams from other start-ups (context of box 2 in the figure).

Figure 3.1: Methodological concerns affecting implementation of MDE as a Service

Between 2008 and 2010, Adapit planned and developed an approach for automated design. It includes a tool called MockupToME and other DSLs in a methodology called MockupToME Method (BASSO *et al.*, 2016d). It is the result of three years of industrial innovation, planned exclusively for the application of MDE in target software projects for web information system. In 2010, it was implemented a feasibility study for MDE as a Service, by introducing this new method to other start-up, referred as "Company A", thus characterizing a scenario for inter-organizational reuse (BASSO *et al.*, 2017a).

Differently from Adapit, Company A adopts Scrum (MOE *et al.*, 2010) as the reference model for the software development process and has some preferences for tool support different from those used by Adapit teams. Likewise, MDE resources were adapted for the target context using a set of toolboxes for tool chain (BASSO *et al.*, 2014a), analyzing issues associated with this specific reference model in combination with MDE (BASSO *et al.*, 2015b).

3.1.1 Goal

We aim at reporting methodological concerns for implementation of MDE as a Service, thus investigating the following research question: **Q1: Which are the relevant aspects to be concerned with in implementations of MDE as a Service?**

3.1.2 Research Method

We conducted a new longitudinal study by extracting data from two studies (C02 and C04) shown in Table 3.1. These studies are selected because they highlight our concerns for implementations of MDE as a Service less tied to technical issues. Besides, all these studies use data derived from real settings, highlighting many issues affecting acceptance of MDE tools and methods in industrial contexts as well

as some experiences that gave good results. However, since our goal is to report on methodological concerns rather than toolbox concerns, we selected only information that is not dependent from tool demonstration, thus excluding technical concerns discussed in studies C01 and C03.

Table 3.1: Studies for characterization of MDE as a Service

Id	Reference	Title	Challenge Focus	Where?
C01	BASSO *et al.* (2014a)	Supporting Large Scale Model Transformation Reuse	Reuse Mechanisms	ACM SIGPLAN Notices
C02	BASSO *et al.* (2015b)	Combining MDE and Scrum on the Rapid Prototyping of Web Information Systems	Integration in SDP/Teams	IJWET
C03	BASSO *et al.* (2016d)	Automated Design of Multi-layered Web Information Systems	Tool Chain Implementations	JSS
C04	BASSO *et al.* (2017a)	Building the Foundations for "MDE as Service"	Inter-Organizational Reuse	IET Software

For more detail, the following content is found in each study:

- **C01 -** presents a report about customizations introduced in model transformation components in support for the development of three web information system projects;
- **C02 -** introduces challenges for integration of MDE Toolboxes in agile software processes;
- **C03 -** details the MockupToME Method, an automated design approach for web information systems assisted by MDE Toolboxes, i.e., it is shown the final result from a long tool chain configuration process, and;
- **C04 -** provides foundations for MDE as a Service through toolboxes built in our research group.

3.1.3 Analysis

This section provides answers for the research question **Q1: Which are the relevant aspects to be concerned with in implementations of MDE as a Service?**

Lack of studies combining MDE and arbitrary reference models for software development process in practice. The body of knowledge concerning practical issues in implementation of MDE as a Service is poorly discussed (BASSO *et al.*, 2014a, 2015b, 2016d, 2017a; BRAMBILLA and FRATERNALI, 2014; MOHAGHEGHI *et al.*, 2009, 2013; MONTEIRO *et al.*, 2014a,b). This finding corroborates with HEBIG and BENDRAOU (2014), who claimed that future studies in Software Engineering should evaluate the impact that introduction of MDE causes on reference process models in general, in special evaluating the daily practice of teams. Thus, our analysis concludes that tool chain in MDE involving reference models is an open topic for research.

The literature of the area lacks information on how to introduce MDE in specific contexts. Although many works are essential to understand threats that involve MDE adoption (HUTCHINSON *et al.*, 2011; PETRE, 2013; WHITTLE *et al.*, 2015), they only present a superficial analysis of the problems. A limitation of these studies is the lack of focus on specific application domain, such as web information systems or embedded systems. The quick development of prototypes could be useful in these domains. Thus, in order to better understand issues that hamper MDE adoption in industry, empirical studies should be directed for specific contexts.

Which are the incompatibilities in terms of tool between MDE and Agile? The subjectivity of the information available in the literature of the area makes very hard to understand whether a model-based tool is incompatible with agility principles. Most of information that we have found related to Agile principles resembles general recommendations of software development, which is not useful for one who needs to use existing MDE resources in the context of a Scrum-based software project. Several assumptions have to be defined, which is a difficult decision for the service provider, since no one knows exactly how to deal with these approaches together. In (BASSO *et al.*, 2015b), we have invested big effort to understand incompatibilities between MDE and Scrum and at the end of our analysis we have realized that the unique relevant issue to the combination of these approaches is the need of Sprints with short timescale. Our analysis concludes that the body of knowledge for decision making in this matter is poor, thus implementations for MDE as a Service are ad-hoc.

Which are the requirements for "agile tools" in the context of MDE? Which design tools are more suitable for agile teams? In which contexts? With which goals? These questions should be explored in empirical studies conducted in industry. The literature presents several proposals for MDE and rapid application prototyping, but no criteria that could help Software Engineers to choose the best option for agile contexts is provided. This limitation makes very difficult to match the theory related to Agile Methods and MDE with practice needs of industry. As a result of the lack of requirements for agile tools, the decision making process on which tools and tasks to include in a defined process for a target company is still ad-hoc.

The "good" and "bad" on the combination of MDE and Agile should be associated with a context. In our view, answers for questions like the above ones cannot be generalized to all companies, processes and team configurations. To be valuable for the Software Engineering practice, they have to be answered for each specific context (e.g., big, small or start-up companies, teams with different configurations, etc). Accordingly, we have reported in our previous contributions

(BASSO *et al.*, 2014a, 2016d, 2017a) the importance of target context for choosing a suitable DSL to start the MDE-based process. Therefore, instead of making general assumptions on the applicability of a process model, methodologies and tools, Software Engineers should direct their observations for specific contexts.

A step back regarding to the representation of workflows in the FOMDA DSL: a domain specific language for representation of MDE Settings proposed in 2006 in the author's Master's dissertation (BASSO, 2006). FOMDA DSL was tested and improved in industrial settings until 2011 (BASSO *et al.*, 2014a, 2017a), allowing software engineers to instantiate tool chains through representations of four concepts: software product lines, model transformation chain, component based development and adaptive test cases. In the older version (2006), we thought that the use of a workflow designed with the FOMDA UML Profile (BASSO *et al.*, 2006, 2007) would be a good representation for tool chain components. However, due to the complexity required for designing of a cross-application domain model, the practice suggested the opposite. The specificity of MDE components implies on the use of several UML annotations, which is bad for the understanding of a workflow due to the design pollution. In other words, with the FOMDA UML Profile it seems that the essence of workflow representation has been lost in detriment of a more technical-level representation. Thus, we did not include workflows in recent versions of FOMDA DSL after 2009 (BASSO *et al.*, 2009).

A research direction for integration with Software Development Processes (SDP). One of our goals with the FOMDA UML Profile was to represent the specificity of MDE resources in SDPs. This has been performed by some other researches in the area (MACIEL *et al.*, 2013), which conclude that the inclusion of representations for model, metamodels, and transformations in a process modeling language called SPEM is good from the point of view of process engineers. However, our experiences with the FOMDA UML Profile, with similar representation of workflows, suggest the opposite. Besides, we have been noticing that MDE resources are agnostic to SDPs. These divergent positions raise two questions that should be investigated: 1) Whether the integration of technical-level information in process modeling languages is good for reuse? 2) Due to issues previously discussed, which implied in a step back for the inclusion of our technicalities in workflows, an extension of a process modeling language is good for the practice?

For instance, we are considering a less intrusive approach for introducing automatically MDE resources in SDPs specifications through tailoring rules for BPMN (PILLAT *et al.*, 2015).

3.1.4 Final Remarks

Through these four characterization studies, we present a contribution for the theory and practice in implementation for MDE as a Service. However, all these reported issues are from previous practices, thus could be related only with a personal perspective of this scenario. In order to remove this bias, we executed structured reviews, summarized latter in Section 3.4, to find out whether other reports are concerned with the same issues in implementations of this opportunity. Next, we present a structured review of type mapping study discussing this scenario.

3.2 Implementation Concerns

MDE achieved a certain maturity in practice and research (MUSSBACHER *et al.*, 2014; WHITTLE *et al.*, 2015), leading software engineers to the development of several tools, languages and methodologies for specific needs (PETRE, 2013). Although the advances, tool chain remains a challenge. MOHAGHEGHI *et al.* (2013) claim that:

> *"developing a MDE tool chain requires high expertise and an investment of effort since no platform works "out-of-the-box"... Model-driven engineering has not been an out-of-the-box solution for any of the cases discussed in this paper... This fact is both the strength and the weakness of the MDE approach: developing the environment requires high expertise and is costly, which does not pay off for small or single projects. However, once such a development environment is created, it has the potential to be easier to use than generic modelling tools and to save effort by automation."*

This is a bit surprising coming from authors who reportedly have been using tools built on Model-Driven Architecture (MDA) (KLEPPE *et al.*, 2003) standard and Eclipse Modeling Framework (EMF) (STEINBERG *et al.*, 2008) ecosystems. Paradoxically, they should not observe these tool chain problems because they use an interoperable set of tools, but they did have an issue with integration. Since we have been observing the same issues (BASSO *et al.*, 2016d, 2017a), an open question is why integration is so hard (ZAKHEIM, 2017)?

In this section, we want to understand what affects representations of tool chain, i.e., why MDE has not been an out-of-the-box solution. For example, in the study that culminated into this section, it is concluded that an integrated reuse mechanism and approach for the management of cross-application domains is essential to a successful implementation of MDEaaS (BASSO *et al.*, 2017a). However, some limitations imposed difficulties for the business feasibility, thus opening window for

substantial new contributions in the area. For example, current tool chains are more like an orchestration of several systems for MDE (AHO *et al.*, 2009; BATORY *et al.*, 2013b; HEBIG and BENDRAOU, 2014; LAFI *et al.*, 2011; VARA *et al.*, 2014) than the use of a unique tool support (GRUHN, 2002; LININGTON, 2005; WASSERMAN, 1990). This change of paradigm is observed in first contributions (ANDERSSON and HST, 2008; BECKER *et al.*, 2002; SOUZA *et al.*, 2007; STARY, 2000; STOCQ and VANDERDONCKT, 2004), where a model is designed with a unique design language and transformed by a unique script for model transformation. In current approaches, one can observe that MDE-based processes need to orchestrate several tools with some (semi-)automatic Software Engineering (SE) tasks that associate systems for MDE (BATORY *et al.*, 2013b; HEBIG and BENDRAOU, 2014; VARA *et al.*, 2014).

The interest in tool chain research has increased in recent years (AHO *et al.*, 2009; BIEHL *et al.*, 2014; BRUNELIÈRE *et al.*, 2010; ELAASAR and NEAL, 2013; FRANTZ and CORCHUELO, 2012; ZHANG, 2015). Despite the interest, we lack characterization of the area. A first threat is that the execution of an approach for tool chain may include many toolboxes. Besides, the context of system engineering, as the one experienced at Adapit and discussed in Section 3.1, is not unique that can be target of an implementation of MDE as a Service. However, many other tasks performed through MDE Toolboxes are also important, such as those under the umbrella of process engineering (PILLAT *et al.*, 2015).

In this direction, we advocate the need for characterization to conduct our analysis in MDE as a Service, as well as to highlight contributions in the state of the art for tool chain. For that, we organized our research as described in the following subsections.

3.2.1 Goal

In order to provide an analysis of the current scenario for implementation of coopetition in MDE as a Service, our goal is to find what affects a MDE tool chain instantiation. In other words, we want to identify the possible MDE Artifacts and Settings to be used in instances of tool chain as well as what is considered in this scenario for tool chain in Software Engineering research. In this direction, we conducted a structured review of type mapping study to answer the following research question:

Q2: Which are the relevant aspects to be concerned with in integration of MDE Artifacts and Settings in the context of MDE as a Service?

3.2.2 Research Method

To achieve this goal, we performed a structured review of type mapping study using a snowballing protocol (PETERSEN *et al.*, 2008). This mapping study counts on a total of 108 researchers registered in `researchgate.net`, whose publications have been monitored by us since March 2012. In the following, we describe the result of our findings.

3.2.3 Analysis

This section answers **Q2: Which are the relevant aspects to be concerned with in the integration of MDE Artifacts and Settings in the context of MDE as a Service?**. Next, we describe some of the main terms used by tool chain approaches.

Standard Transformation Lifecycles

Standard transformation lifecycles are found in approaches for management of transformation lifecycles built on standard tool chain constructors. For example, these lifecycles include the MDA (KLEPPE *et al.*, 2003), common approach adopted for model transformation lifecycle management. It was proposed in 2001 by the Object-Management Group (OMG) in order to standardize model-based approaches. MDA is an initiative to help software engineers in the management of software development complexity using models at higher levels of abstraction. It can be understood as a "guideline" for the construction of MDE-based processes from some technologies classified as standards, such as XMI, OCL and UML. It does not define explicitly the diagrams that should be used and neither the required transformations among the three different kinds of models that it defines as views: Computational Independent Model (CIM), Platform Independent Model (PIM), and Platform Specific Model (PSM).

Eclipse Modeling Framework (EMF) (STEINBERG *et al.*, 2008) is a toolbox for metamodeling. Model transformation lifecycles built on EMF include DSLs interoperable through XMI. Since these DSLs are represented in Ecore metamodels, they can also be composed (FARIAS, 2010). These are well established standards for metamodeling and tool chains typically does not offer threats for coopetition. For example, EMF includes a kernel of tools called Model Development Tools (MDT) project[1], which is an ecosystem of Eclipse plugins that can be easily instantiated in a tool chain. This is a success case in the area enabling coopetition through a common platform.

[1]MDT <https://eclipse.org/modeling/mdt/>

However, to all rules there are exceptions. Industry adopts several constructors for MDE Artifacts and Settings (COMBEMALE *et al.*, 2014; LIEBEL *et al.*, 2014; MUSSBACHER *et al.*, 2014; WHITTLE *et al.*, 2015), imposing threats for the implementation of coopetition in the area. In the following section, we discuss this heterogeneous and hybrid alternatives.

Hybrid Transformation Lifecycles

Hybrid transformation lifecycles characterize heterogeneous constructions of tool chains in MDE-based processes. For example, including diverse types of model transformations (LÚCIO *et al.*, 2014) and settings non-standard or ad-hoc approaches for tool chain that manages model-based operations with simple solutions for modeling, such as databases (BATORY *et al.*, 2013b), JSON (IZQUIERDO and CABOT, 2013), and even Excel Spreadsheets (CUNHA *et al.*, 2016; RESCHENHOFER and MATTHES, 2015). Differently, MDA and EMF lifecycles are restricted to adopt their standards in tool chains for MDE.

The following features characterize existing contributions in transformation lifecycles composed of:

1. Models represented with different levels of abstraction according to different metamodels (VOELTER, 2009) and technologies that support platform-dependent implementation levels (e.g., Java annotations (BASSO *et al.*, 2014c), Eclipse Java development tools (JDT) (TURNER and CHAE, 2010) and database perspectives (BATORY *et al.*, 2013b; QUERCINI and REYNAUD, 2013));

2. Models can be in conformity with Abstract Syntax Trees (AST) (HEIJSTEK *et al.*, 2011), such as exemplified in (FORWARD *et al.*, 2012; OLIVEIRA *et al.*, 2011). Thus, textual meta-generators (NEUBAUER *et al.*, 2015) such as ANTLR [2] and XText [3], are characterized as meta-definitions;

3. Meta-definitions (known as tools for construction of metamodels) are models too and can be expressed with diverse technologies. For example, those illustrated in level `PIM'2` of Figure 2.3 characterize meta-metamodels built on EMF (STEINBERG *et al.*, 2008), AToM3 (DE LARA and VANGHELUWE, 2002) and Meta-Edit+ (KELLY and TOLVANEN, 2008). Other exotic approaches for meta-definitions include entity-relationship from database perspectives (BATORY *et al.*, 2013b), native JDT-based meta-models (BASSO

[2]ANTLR- <http://www.antlr.org/>
[3]XText- <https://eclipse.org/Xtext/>

et al., 2014c) and XML Schemas (e.g., those used for construction of DSLs such as UsiXML[4] and BPMNt (PILLAT *et al.*, 2015));

4. Model transformation components and sub-components built in diverse types of the aforementioned meta-definitions (LÚCIO *et al.*, 2014); and

5. Toolboxes of any nature that can be used in a MDE context (ELAASAR and NEAL, 2013; ZHANG and MOLLER-PEDERSEN, 2014).

These different perspectives for MDE Artifacts and Settings are threats for integration in tool chain.

MDE Artifacts in Hybrid Transformation Lifecycles

The best definition we have found in the literature to express how diverse and hybrid are MDE Artifacts is presented by France:

> *"A simple artifact is the smallest unit of information ... a set of tightly coupled elements ... Examples of simple artifacts are UML class descriptions, UML relationships, Java programs, metamodels, test cases, and method descriptions. Each artifact has a type that contains metadata about the artifact and that specifies the kinds of manipulations that can be carried out on the artifact. The kinds of manipulations supported by an artifact can be described in terms of an interface that specifies allowable operations in terms of their signatures and constraints on their behavior... An artifact type can also specify data integrity and access control rules that are applicable to all artifacts of the type. "* (FRANCE *et al.*, 2006)

According to the aforementioned definition, a tool chain is a complex MDE Artifact. It is also characterized as an automated process (LIEBEL *et al.*, 2014), where stakeholders make use of some toolboxes and models in different software development phases. For this reason, tool chains for MDE are even more complex to represent than those for Application Lifecycle Management (ALM) tools (ZAKHEIM, 2017; ZHANG and MOLLER-PEDERSEN, 2013).

For example, assume a request for chaining the unique model transformation illustrated in Figure 3.2, where from a model "Model 1" one should generate a model "Model 2". These models are artifacts for system engineering with elements annotated for Object-Relational Mapping (ORM) (BURKE and MONSON-HAEFEL, 2006), an important concept for data persistence used in the development of information systems. Assume that a tool for applying ORM in class diagrams is integrated

[4]UsiXML-<http://www.usixml.org/>

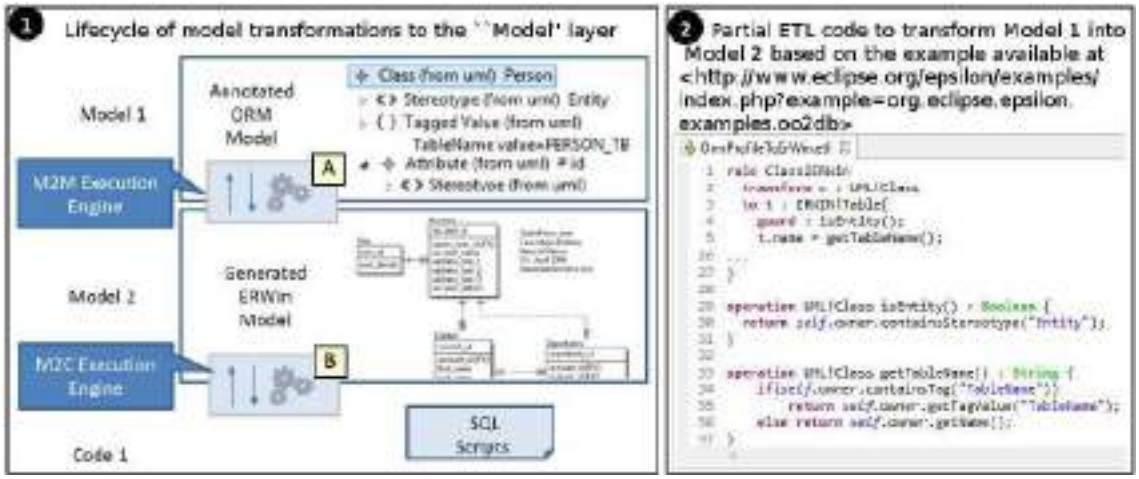


Figure 3.2: A model layer annotated with ORM Profile.

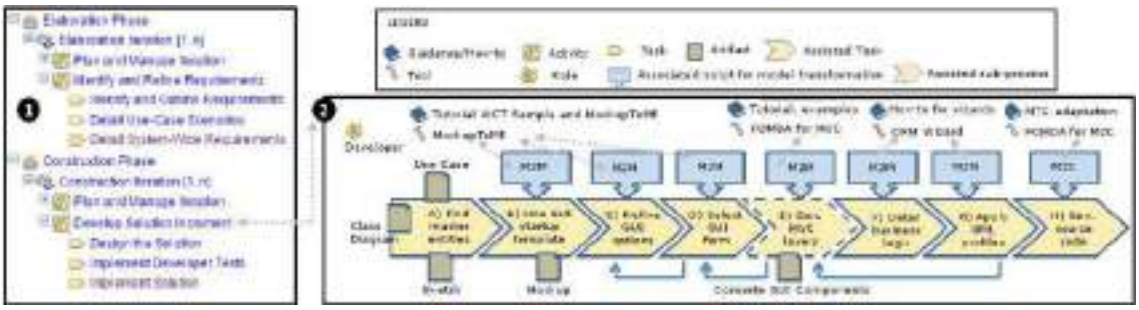


Figure 3.3: Method with integrated transformations.

to the task "G: Apply UML Profiles" of the method shown in Figure 3.3 (2) and this method should be integrated in an activity called "Develop Solution Increment" from the process model shown in Figure 3.3 (1). In this simple example, a software engineer needs to chain at least three tools:

1. for execution of script developed with ETL (JOUAULT *et al.*, 2010);
2. for execution of method represented in SPEM (MACIEL *et al.*, 2013), and;
3. for enactment of process represented in EPF Composer (STEINBERG *et al.*, 2008).

Therefore, even considering the integration of a simple transformation task, ability for acknowledging and representing technicalities in three program execution mechanisms and/or languages is necessary.

The practice related to tool chain usually considers tools non-based on MDE (ZAKHEIM, 2017). ZAKHEIM (2017) claims that software engineers know how to chain tools, but it is still a hard task even when using the best option in tool support. In MDE, the conclusion is the same (LIEBEL *et al.*, 2014). For example, it is possible to chain tools using model transformation engines (BÉZIVIN, 2005),

such as the Epsilon Eclipse plugin (KOLOVOS *et al.*, 208). This is the most common approach (KUSEL *et al.*, 2015). In this sense, due to the use of well formed sequences of model, metamodel and transformations, the most appropriate term is Model Transformation Chain (MTC) (ETIEN *et al.*, 2012), a particular case of tool chain.

However, it is important to highlight that there is no "silver bullet" in tool chain. For example, some approaches proposed the use of software process specifications (MACIEL *et al.*, 2013) and execution of model transformations through *Process*-Centered *Software Engineering Environments* (PSEEs) (GRUHN, 2002), which have been observed recently as a way to automate MDE-based processes (POLGÁR *et al.*, 2009). MATINNEJAD and RAMSIN (2012) surveyed benefits and drawbacks of PSEEs and reasoned that, although many free and commercial solutions exist, few cases of adoption in industry are reported. Earlier, GRUHN (2002) pointed to PSEEs as a solution to several execution environments, used to automate small parts of the software process, by a single environment. Although this high availability of tool support for tool chain, instead of modern tools proposed by the literature, BATORY *et al.* (2013b) concluded that some software engineering students still prefer to use an Integrated Development Environment (IDE) for tool chain. This suggests the lack of agreement of tool chain representations. This way, ZAKHEIM (2017) suggests that software engineers should make explicit data for integration, in order to facilitate identification of the most appropriate integration tool.

General Purposes MDE Toolboxes

Below we discuss some elements associated with the adoption of miscellaneous toolboxes that aim at providing facilities for introducing MDE Artifacts in target contexts.

Maturity in toolboxes supporting MDE tool chain. Some challenges for MDE tool chain were introduced in 2002 by MELLOR (2002), who highlighted the importance of extensible model tool chains to reach model-based approaches. He claimed that an appropriately specified kernel for MDE would enable an increasingly powerful chain of tools including:

> "***Model builders.*** *Providing graphical input of models;* ***Model verifiers.*** *Interpreting a model with real values so users can determine whether the behavior is correct;* ***Model compilers.*** *Compiling models onto diverse platforms;* ***Model debuggers.*** *Executing compiled code so developers see the code compiled from the model in action;* ***Model analyzers.*** *Finding paths through the models and unreachable states; and*

Model testers. *Generating and running test cases for models."* (MELLOR, 2002)

However, this definition is over-dated, requiring a panoramic view on topics that need to be investigated in tool chain research.

Wide availability of toolboxes for construction of MDE Artifacts as "model management tools" (MUSSBACHER *et al.*, 2014). Several tools have been proposed along the years to help developers in producing resources for MDE (e.g., model transformations, models, metamodels and APIs), some surveyed by JAKUMEIT *et al.* (2014). Current MDE tools are based on well established concepts for transformation and metamodeling (KELLY and TOLVANEN, 2008) and most of them are implemented with Eclipse Modeling Framework (EMF) (STEINBERG *et al.*, 2008). Model transformation languages such as ATL (BÉZIVIN, 2005), QVT[5] and ETL (KOLOVOS *et al.*, 2006) are available in MDT to support the development and execution of model-to-model transformations. Others such as MOF to Text (OMG, 2008) and MOF Script[6] are available in MDT to support the development and execution of model-to-code transformations.

Wide availability of toolboxes for generation of DSLs (meta-generators). Most of UML tools have support for UML Profiles, where this language is extended with tags and stereotypes to add a different semantic to the annotated model element. Because UML Profile extensions are conservative, the metamodel remains the same. On the other hand, Domain Specific Modeling Languages (DSMLs) (KELLY and TOLVANEN, 2008) provide a new metamodel that can be generated with Eclipse Modeling Framework (EMF) (STEINBERG *et al.*, 2008) or other approaches for metamodeling such as MetaEdit+ (TOLVANEN, 2016), AToM3 (DE LARA and VANGHELUWE, 2002), Obeo Designer (KOUHEN *et al.*, 2012) and Microsoft DLS Tools (COOK *et al.*, 2007).

Diverse contexts are adopting hybrid metamodels/DSL in hybrid transformation lifecycles. Much of the MDE research effort is directed to development of DSLs and tool support to manage model transformation tasks. In summary, most researches focused on: 1) proposing diverse modeling languages (BÉZIVIN, 2005), techniques and methodologies to represent abstractions (e.g., textual and graphical DSLs, UML Profiles) from specific (SELIC, 2005) or general domains (BATORY *et al.*, 2008); 2) proposing mapping from model representations to other specifications (BECKER *et al.*, 2002), such as BRAMBILLA and FRATERNALI (2014) that reported an industrial practice in which BPMN flows (using a DSL to represent business models) are mapped to graphic interface

[5]QVT <http://www.omg.org/docs/ptc/05-11-01.pdf>
[6]MOFScript <http://eclipse.org/gmt/mofscript/>

flows conforming to WebML (another DSL); 3) demonstrating how models are generated and refined in views through model-to-model transformations (BÉZIVIN, 2005; KOLOVOS *et al.*, 208), how models are used to generate source-code for target platforms through mappings (BECKER *et al.*, 2002)); 4) developing "model management tools" (JAKUMEIT *et al.*, 2014) and; 5) development of underlying frameworks (VARA *et al.*, 2014), whose concepts are usually associated with the term MTC (VANHOOFF *et al.*, 2006), offer support for execution of model transformations developed with more than one language (YIE *et al.*, 2012) and allow composition of model-based operations (ETIEN *et al.*, 2013).

Toolboxes for Management of Evolution

We can mention the following modern toolboxes providing model-based operation support for evolution of models, metamodels, transformations and tool chains:

Support for co-evolution of model, metamodel and model transformation components. Few contributions have considered a solution for co-evolution of model transformations, application models and metamodels (CORRÊA *et al.*, 2013; LOPEZ-HERREJON *et al.*, 2010; ROSE *et al.*, 2013). In these contributions, as a metamodel evolves, transformations and application models are automatically changed.

Refactoring of model transformation components. Refactoring of model transformations is inevitable (ALVES *et al.*, 2006; HOLDSCHICK, 2012; WIMMER *et al.*, 2012), since implementation technologies evolve in industrial contexts. In order to quickly adapt model transformation resources for new requests, refactoring of model transformations allows to introduce automatic changes in code/specifications of transformation components. Heterogeneity in these languages also limits existing refactoring approaches.

Traceability of models, meta-models and model transformations. To trace artifacts is important for the maintenance and evolution of software (BRAMBILLA and FRATERNALI, 2014). Recent contributions in tool support allow to trace generated artifacts through the execution of transformations (VARA *et al.*, 2014).

Model transformation reuse. Modern model transformation engines support languages able to manage dynamic execution based on variability (KUSEL *et al.*, 2015).

Model transformation generation. Modern model transformation engines support languages able to represent common concepts from model transformation languages (CUADRADO *et al.*, 2014) for generation of transformations in specific formats.

Model composition. There are several mechanisms and techniques for model

composition, regardless of which model, e.g., for process (MAGDALENO *et al.*, 2015) or for architectural representations of software (FARIAS, 2010). Therefore, research in this topic is mature in theory and practice (GONÇALES *et al.*, 2015).

Toolboxes for Process Engineering

A target context for introduction of MDE may be a process. Below we describe three main research topics under investigation in process engineering and MDE.

MDE Toolboxes can be used in any phase of a software development process. Several DSLs and tools have been proposed to support different intents: 1) Loniewski et al. surveyed MDE-based techniques to assist requirement engineering tasks (LONIEWSKI *et al.*, 2010); 2) AKYILDIZ *et al.* (2002); YICK *et al.* (2008) surveyed the development of wireless sensor networks, which include some design tools used in SDP tasks after requirement elicitation such as "design the solution" and "develop solution increment"; 3) Other authors suggested specific tools to design web information systems in preliminary phases of requirement elicitation (MOLINA *et al.*, 2012; RISTIĆ *et al.*, 2012) and; 4) BENAVIDES *et al.* (2010), BERGER *et al.* (2013), CAPILLA *et al.* (2014) and THÜM *et al.* (2014) summarized some solutions for product companies, used in requirement management based on concepts of software product lines, which is a type of model-based approach independent from the application domain.

Several SDP reference models are available for executing MDE-based operations. In order to combine MDE and Agile Methods, some reference models (or process frameworks) such as Agile Model Driven Architecture (AMDA) (AMBLER, 2015) and Feature Driven Development (FDD) (CHOWDHURY and HUDA, 2011) have been proposed and applied for software development. Some interesting works combining Scrum and MDE can also be mentioned, such as from KULKARNI *et al.* (2011), which proposed a new SDP called Meta-Sprint, and ZHANG and PATEL (2011), which proposed the Agile MDD. Thus, for an initiative towards the implementation of "MDE as a Service", several reference models for SDP could require the integration of also several MDE Artifacts, as suggests some of our previous experiences (BASSO *et al.*, 2015b, 2016d).

Industrial contexts present variability in relation to SDP reference models. Some software process models followed by companies impose difficulties to introduce MDE Artifacts in target contexts (HEBIG and BENDRAOU, 2014). WHITTLE *et al.* (2015) claimed that instead of using a unique SDP reference model to conduct a MDE-based process, demands from the industry present specific contexts, including the use of diverse SDPs and team skills that can present threats to MDE adoption. These factors influence the acceptance of tools for modeling and model transformation (HEBIG and BENDRAOU, 2014; PETRE, 2013),

for example. Thus, some industrial contexts require integration and adaptation of arbitrary resources for MDE (tasks, tools and DSLs) into enterprise specific SDPs, which can result in inconsistencies (FUGGETTA and NITTO, 2014). HEBIG and BENDRAOU (2014) claim that future studies in Software Engineering should evaluate the impact that introduction of MDE causes on reference process models in general. Based on these works, a software project conducted with a Scrum-based framework can impose threats to the introduction of MDE, as well as MDE can impose threats to the software project execution.

Adaptation in SDP specifications. JOHNSON *et al.* (2012) claim that meta theories for software engineering are necessary in order to understand the commonalities and differences among software development practices. This knowledge can be instantiated in software processes represented with Essence, a new OMG specification for software process. DOS SANTOS ROCHA and FANTINATO (2013) surveyed some proposals for method engineering based on product line techniques which adapt process components for diverse contexts in software development companies. These proposals are also called Software Process Lines (SPrL) (OLIVEIRA JUNIOR *et al.*, 2013), implemented with tools such as vSPEM (MARTINEZ-RUIZ *et al.*, 2011). Other tool support closer to composition through model-to-model transformations are based on tailoring rules such as BPMNt (PILLAT *et al.*, 2015) and CASPER (ALEGRÍA *et al.*, 2011), or heavy-weight implementations through EPF composer (KROLL and MACISAAC, 2006).

Toolboxes for MDE Artifact Engineering

The literature of the area lacks a mapping of representations for MDE Toolboxes in artifact engineering, characterizing a relevant topic for research and practice in MDE.

Fragmentation and Composition of MDE Artifacts. For companies that need to tailor model transformations for different software projects (legacy and modern), managing implementation variability in code generators or model-to-model transformations is crucial to the business. This would allow to generate new functionalities as increments for legacy systems (HEBIG, 2014), using model transformations for old-fashioned technologies, and developing new ones with modern technologies. Features Model (FM) (KANG *et al.*, 1990) is used in practice to automatically cut and merge pieces of the model transformations, a practice known as factorization (ARANEGA *et al.*, 2012a). Therefore, it is a good solution to adapt this type of model transformation component.

Variability in MDE Settings and Model Transformations. These concepts were explored in 2012 in model transformation chains (ARANEGA *et al.*, 2012a), in 2013 (HEBIG *et al.*, 2013) and 2014 (CUADRADO *et al.*, 2014) consid-

ering concepts of Component Based Development (CBD). Currently, the widely used notation to formally specify variability is the Features Model (FM) (THÜM *et al.*, 2014). Also referred in regards to transformation components as PDM - Platform Domain Model (TEKINERDOĞAN *et al.*, 2005), Features Model is useful to define system characteristics (functional, architectural, technological, mixed, etc.) of a particular domain. Moreover, it is also possible to use feature relationships to configure variability and commonality in dynamic execution environments. Thus, the Features Model can be used to specify variability required in model transformations and tool chains.

Semantics for Model Transformation. In order to provide semantics for model transformations, some authors recommend classifications (LÚCIO *et al.*, 2014) and use of a fourth view for transformation description (WILLINK, 2003). MDE Settings, therefore, can also present any semantics associated with a model transformation component. For example, to transform a PIM into a PSM, it is necessary to identify characteristics such as APIs and design patterns that will define which component should be used in a model transformation lyfecycle. These characteristics are MDE Settings, associated with arrows in Figure 2.3, allowing the management of a lifecycle through DSLs constructed with metamodels that support semantics and syntax for selection of components.

Generation of Target Representations for MDE Settings. A possible solution for integration of model-based operations in contexts is to generate a setting (BASSO *et al.*, 2014e; OLIVEIRA *et al.*, 2011). Aiming at generating a setting to support the execution of consecutive model transformations, several proposals (ALVAREZ and CASALLAS, 2013; ARANEGA *et al.*, 2012a,b; ASZTALOS *et al.*, 2011; BASCIANI *et al.*, 2014b; BENDRAOU *et al.*, 2008; BIEHL *et al.*, 2014; CASTELLANOS *et al.*, 2014; CUADRADO *et al.*, 2014; ETIEN *et al.*, 2012; KÜSTER *et al.*, 2009; LUCAS *et al.*, 2017; LÚCIO *et al.*, 2013; POLGÁR *et al.*, 2009; ROYCHOUDHURY *et al.*, 2011; VANHOOFF *et al.*, 2006; YIE *et al.*, 2012; ZHANG and MOLLER-PEDERSEN, 2013) tackled issues associated with representation for tool chain. This allows one representing settings through DSLs developed with different intents.

3.2.4 Final Remarks

The state of the art presents many MDE Artifact as options for implementation of MDE-based processes. Some artifacts are complementary and others are overlapping, thus configuring coopetition opportunities in scenarios for MDE as a Service. Next, we present some opportunities for coopetition and limitations in our previous work to meet the new needs introduced by this new scenario in the context of MDE

as a Service.

3.3 Cooperation and Competition Concerns

The importance of tool chain approaches for the software development was recently discussed by FUGGETTA and NITTO (2014), highlighting a trend for cooperative systems that assist Software Engineering (SE) tasks in diverse software development phases. They agreed that MDE is an important paradigm to accomplish this trend. Previously, BOEHM (2006) also highlighted the need of research in this direction, arguing that to compete, adapt, and survive, software development companies will depend on the ability to integrate some systems into global reuse scenarios made of Systems of Systems (SOS) (NETO *et al.*, 2014). Although a big effort is invested in this matter, 11 years later, ZAKHEIM (2017) claims that it is still observable a difficulty to make this view of the future a reality in industry, thus leaving room for substantial improvement.

This trend is of special interest for the MDE context (HEBIG and BENDRAOU, 2014; MUSSBACHER *et al.*, 2014). However, as observed in Section 3.2, a first issue to apply this vision for tool chain is that MDE is immerse on a chaos of possibilities (MUSSBACHER *et al.*, 2014). This includes technical options to implement a MDE-based process as well as methodological and ideological concerns (PETRE, 2013). Usually, things do not match semantically and syntactically (LIEBEL *et al.*, 2014). Companies change their practices, processes and underlying implementation architectures, imposing threats for coopetition in the area. For this reason, MOHAGHEGHI *et al.* (2013) claim that there is no "out-of-the-box" solution in MDE, which makes costly a process to instantiate tool chains.

3.3.1 Goal

We believe that coopetition is the key to reduce cost in MDE as a Service, as illustrated in Figure 3.1 (A). In other words, we can collaborate and compete by analyzing the best third-party options for MDE Artifact to introduce in inter-organizational contexts. This way, the following research question is investigated: **Q3: Which are the opportunities for coopetition in the context of MDE as a Service?**

3.3.2 Research Method

In order to answer this question, we first conducted an ad-hoc literature review. Then we conducted five structured mapping studies, extracting from these studies a set of works in the context of coopetition. First we present a panoramic view of this scenario, where resources developed for MDE (e.g., model transformations,

DSLs and transformation tools) are introduced in different contexts. This is not easy and requires a set of techniques and tool support for reuse that makes the configuration of resources for MDE flexible. An analysis of the target context is carried out, highlighting which resources for MDE are used in the development of a specific software project, e.g., selecting an appropriate DSL to be used in the development of web information systems. It is also important to consider the know how of teams to support the design and development tasks, which may imply on the use of different frameworks, processes and technologies. In a second moment, we exposed some opportunities that we found from platforms for MDE Artifacts.

3.3.3 Analysis

In MDE as a Service, software engineers need to find the right options for each context (WHITTLE *et al.*, 2015). Its execution involves an analysis of which artifacts fit the technical requirements of the contexts, as well as which artifacts are feasible in "philosophical" or "social" matters. For example, in our experiences (BASSO *et al.*, 2015b), principles of agile teams influence the adoption of MDE tools and vice-versa. WHITTLE *et al.* (2015) has the same conclusion, which means that it is important to consider social elements from software development contexts before establishing tool chains. Since high-costs involved in this process (MOHAGHEGHI *et al.*, 2013) cannot be ignored in MDE as a Service, the business feasibility must be considered too (BASSO *et al.*, 2017a).

For these reasons, we stated that MDE as a Service can be considered from the perspective of software ecosystems (SECO) (BOSCH, 2009), integrating these knowledge areas. To Jansen et al., a SECO is a unit of business where a common technological platform for services and software allows to connect resources, information and artifacts (JANSEN *et al.*, 2009). Although ecosystems gained attention from research in recent years (BOSCH, 2009; DOS SANTOS *et al.*, 2013; FUGGETTA and NITTO, 2014), existing work does not identify issues and opportunities for the implementation of cooperation and competition (coopetition) in MDE as a Service.

According to RITALA *et al.* (2014), coopetition benefits the leader of a market and its concurrent, promoting increases in sales in the case of Amazon.com. Although coopetition is an interesting topic for investigation in the business world, so far, the MDE community has ignored the implementation of approaches for coopetition. Meanwhile, approaches for MDE as a Service (MOHAGHEGHI *et al.*, 2013; MONTEIRO *et al.*, 2014a) could benefit from coopetition, which highlights the relevance of software ecosystems research and systems-of-systems to the MDE context.

Competition Concerns

Considering competition concerns, this section answers **Q3: Which are the opportunities for coopetition in the context of MDE as a Service?**

For implementation of coopetition in MDE as a Service, a requirement is to deal with resources for MDE reused in an inter-organizational level (i.e., used by one or more software development companies) (RITALA *et al.*, 2014). We have implemented inter-organizational reuse for MDE Artifacts with the FOMDA DSL (BASSO *et al.*, 2017a), as illustrates Figure 3.4. In order to add flexibility on the generation of adapted tool chains, FOMDA DSL (Figure 3.4.3) is built on concepts for Software Product Line (SPL) and, according to STRUBER and SCHULZ (2016), is one of the options for tool chain representation in the state of the art. Instances of tool chains generated for specific context (Figure 3.4.4) are executed by WCT, a toolbox that includes some elements in support for execution of adaptive model transformations, represented as models in conformity with FOMDA DSL. Finally, these models for tool chain are stored in a local repository (Figure 3.4.2), thus characterizing all the platform used by our previous experiences in implementation of MDE as a Service.

Although FOMDA DSL was essential in adapting tool chains for specific software projects, so far, we were unable to implement coopetition in MDE as a Service by means of this representation language. For example, in BASSO *et al.* (2014e) we conducted a case study by transforming representations in conformity with the FOMDA DSL to representations in conformity with AndroMDA engine[7], which is adopted by the MDArte project (MONTEIRO *et al.*, 2014b) as the core model transformation engine. In the end of this study, we concluded that generative techniques, as transformations, is benefited from cooperation between companies by means of automated integration. This study also introduced a new issue for coopetition: a combinatorial explosion. Due to different representations adopted by each possible combination for transformation, it is necessary to bring the information in accordance with FOMDA DSL to other proposals. Thus, we reasoned that a common representation language would reduce the number of combinations to a linear issue rather than quadratic.

Another issue that we noticed is the lack of elements that could promote coopetition in this scenario. For this reason, BOSCH (2009) makes a distinction between Software Ecosystems and regular Software Product Lines approaches, claiming that when a SPL extends the organizational boundary (i.e., intra-organizational), then a SECO is established to manage inter-organizational resources.

In order to understand why the state of the art in integration is limited for

[7]AndroMDA - <https://www.andromda.org/> (last access on September 1st, 2017)

coopetition, Figure 3.4 illustrates two scenarios for implementation of MDE as a Service:

1. *Current scenario - Homogeneous* - In (2 to 4) is an illustration of implementation considering homogeneous representations for MDE Artifacts and Settings (the state of the art);

2. *Coopetition scenario - Heterogeneous* - Some other implementation options are illustrated in the borders (5 to 7).

This scenario therefore characterizes MDE Resources introduced for inter-organizational. Surveys on the MDE adoption have been reporting that several challenges still hamper these initiatives (AGNER *et al.*, 2013; WHITTLE *et al.*, 2013). For example, as illustrated in Figure 3.1, the diversity of representations tied to MDE resources, as found in requests in this scenario for MDE as a Service and discussed previously in a mapping study, imposes difficulties for tool integration.

In this regard, the implementation of coopetition requires collaboration among different representations outside of the box (Figure 3.4.5, 6 and 7). For these complex scenarios, the success is dependent on software engineer's ability to understand what used when configuring tool chains for specific demands, which change in each target context (WHITTLE *et al.*, 2015). Currently, this implies on the development of new tool chains, toolboxes, DSLs, transformations, and other resources in a quadratic combinatorial explosion. Alternatively, MDE resources may also be searched from free repositories (BASSO, 2015) and/or acquired from a third party stakeholder/company (competitor or collaborator) (DOS SANTOS *et al.*, 2013), but this would require a common representation built on services (RITALA *et al.*, 2014). This adds an extra challenge for tool chain because resources developed by third party are not unknown in a first moment. ZAKHEIM (2017) claims that this integration issue is more related with a business concern involving semantics for artifacts rather than tool support.

Our analysis concludes that we need a common representation in level of assets, as those perspectives promoted by SECO (BOSCH, 2009; SANTOS *et al.*, 2016). Assets should also included structural features for tool chain, such as those used in FOMDA DSL (BASSO *et al.*, 2013a). Otherwise, coopetition will remain a non observable practice and costly in terms of implementation of integrators/connectors in MDE as a Service scenarios.

Cooperation Concerns

Considering cooperation concerns, this section answers **Q3: Which are the opportunities for coopetition in the context of MDE as a Service?**

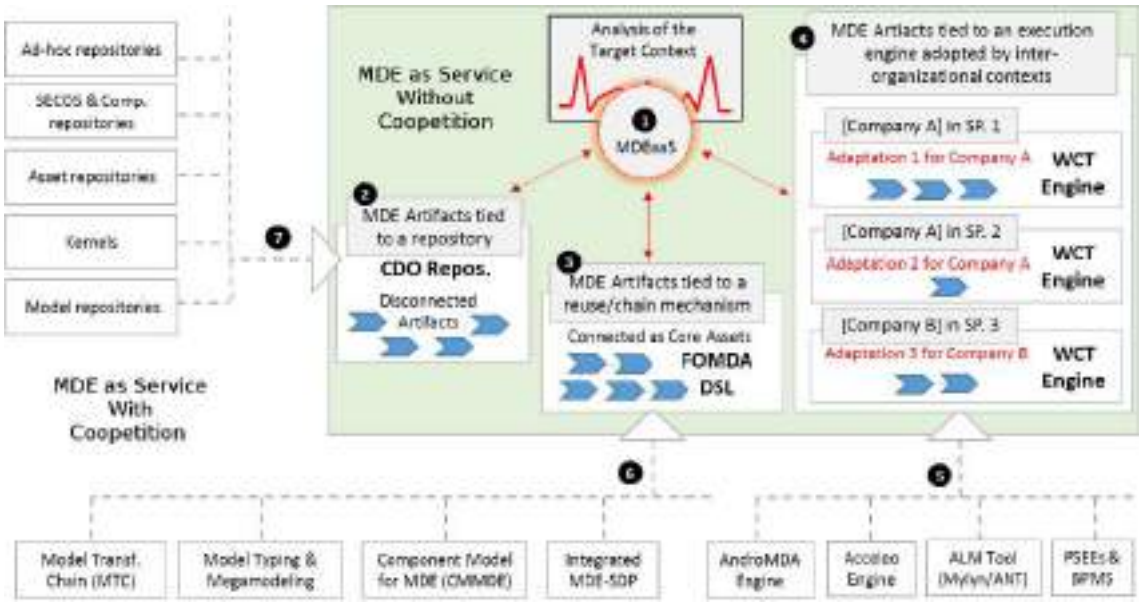


Figure 3.4: Illustration of possible coopetition scenarios in MDEaaS

The following main initiatives for MDE Ecosystems configure opportunities for coopetition in the context of MDE as a Service: the Repository for Model-Driven Development (ReMoDD) (FRANCE *et al.*, 2007), Sharing Hosted Autonomous Research Environments[8] (SHARE) (GORP and MAZANEK, 2011), Globalization of Domain Specific Language (GEMOC) (COMBEMALE *et al.*, 2014), and Software Engineering Methods and Theory (SEMAT) (JACOBSON *et al.*, 2012). Researchers associated with these platforms are making an effort to build a basis of reusable MDE Artifacts that should not be ignored.

MDE Knowledge Base (KB). ReMoDD (FRANCE *et al.*, 2007) is a repository that shares some didactic material for MDE published in some conferences such as MODELS, ECMFA, etc. Most information is available in documents, papers, tutorials, models, metamodels and transformations. In (MUSSBACHER *et al.*, 2014), the authors claimed that this KB will centralize good practices, but that the lack of critical mass imposes difficulties since we have no habit to share information in the area. ReMoDD, therefore, is a repository in operation that can be important to help in reducing the learning curve, an issue that we have observed when introducing MDE in target contexts.

Globalization of DSLs. In order to share resources for MDE, it is important to ensure that eventual compositions in MTCs are valid. The GEMOC initiative is an effort to ensure that technicalities from MDE will be interchangeable in practice (COMBEMALE *et al.*, 2014). In other words, GEMOC will enable a collaborative scenario for MDE considering heterogeneous inter-organizational contexts. This is important for MDE as a Service, since GEMOC can help in reducing the costs

[8]SHARE - http://is.ieis.tue.nl/staff/pvgorp/share/

to introduce MDE in practice. Our analysis concludes that GEMOC is focused in DSLs issues, thus characterizing coopetition opportunities for design and refinement tools.

Knowledge Base for Processes. SEMAT (JOHNSON *et al.*, 2012) is an initiative to provide a knowledge base in Software Engineering related to process models. This is important because some companies target for MDE adoption have not defined their SDP, making costly the analysis of the target context. SEMAT can help in reducing costs through information about processes. Besides, SEMAT uses Essence (OMG, 2015) as a core representation language, which can be used in the context of MDE to automatically integrate technical resources for MDE with target process models represented with Essence. Our analysis concludes that SEMAT platform is important for coopetition opportunities in terms of representations for processes, which can be instantiated for requests from target contexts for tool chain in scenarios that need automation of processes with Business Process Management Systems (BPMS) (PILLAT *et al.*, 2013) or with a Process-Centered Software Engineering Environment (PSEE) (MACIEL *et al.*, 2013).

MDE Forge. ROCCO *et al.* (2015) claim that MDEForge is a repository for MDE Artifacts such as tools, models, metamodels and transformations. Recent contributions associated with MDEForge include: 1) automatic chaining of model transformations (BASCIANI *et al.*, 2014b); 2) cloud computing implementations (BASCIANI *et al.*, 2014a); 3) repository implementations for evolution of models, metamodels and transformations (ROCCO *et al.*, 2014), and; 4) collaborative modeling (ROCCO *et al.*, 2016), or modeling as service. Our analysis concludes that MDEForge is focused in model transformation issues, thus characterizing coopetition opportunities for this type of component.

SHARE. GORP and GREFEN (2012) claim that SHARE is a negotiation platform for MDE Toolboxes, thus with structural features supporting business transactions. Many conferences such as ASE, ECMFA, MODELS, OOPSLA and ICSE recommend the use of SHARE as a platform for tool paper research. This way, SHARE owns many options in tool support for download, some free and other with commercial issues. Our analysis concludes that SHARE characterizes coopetition opportunities in terms of MDE Toolboxes.

3.3.4 Final Remarks

Since reuse allows to reduce costs (KRUEGER, 1992), a global reuse scenario implemented through cooperation and competition could be important for execution of MDE as a Service. This scenario is recently referenced to as coopetition (AXELSSON *et al.*, 2014; PALMQUIST, 2014). Implementing coopetition in MDE as

a Service is a long-term goal. This could open opportunities for new business, reducing costs throughout cooperation in assets for MDE. However, this scenario is still impracticable due to some reasons such as immaturity in processes as well as limitations in tool support for coopetition.

In summary, we found the following opportunities for coopetition that shall be instigated along the next chapters: ReMoDD is sharing diverse MDE Artifacts, SEMAT is sharing methods and process models (JACOBSON *et al.*, 2012), GEMOC is sharing DSLs (COMBEMALE *et al.*, 2015a), MDEForge is sharing model transformations (ROCCO *et al.*, 2016) and SHARE is a business platform promoting the access of Toolboxes (GORP and GREFEN, 2012). These are, therefore, complementary repositories that classify opportunities for coopetition in the area.

Finally, in order to promote coopetition on a perspective of ecosystems, MDE researchers and practitioners could: 1) investigate the applicability of approaches for software ecosystems to promote the reuse of MDE Artifacts, thus helping in the MDE adoption; 2) propose and develop platforms as services for MDE Ecosystems, e.g., finding the requirements for the integration of OSLC (OSLC, 2017a) in this scenario; 3) propose approaches for a network of collaborative services, connecting people, processes, tools and companies on the support for coopetition in MDE as a Service, and; 4) investigate a common representation to enable coopetition through services.

3.4 Representation Concerns

The research community agrees that MDE is in its infancy (MUSSBACHER *et al.*, 2014), thus this is the right moment to build foundations for future implementations through cooperation and coopetition (COMBEMALE *et al.*, 2014). Coopetition requires a common representation (RITALA *et al.*, 2014), an open issue in the MDE research. As shown in Figure 3.5, to contribute to this question, we suggest three preliminary phases for the chain of tools that must be met by this language (BASSO *et al.*, 2013b,c). In the absence of studies concerning preliminary phases for tool chain, this research conceptualized these phases with an ad-hoc strategy, built on our experiences. For this reason, we are not evaluating whether they are complete or methodologically correct for implementation of coopetition. Instead, we are using these phases to find which properties should be considered in RAS++. Thereby, the following phases are considered in our research:

Phase 1 - Asset Specification intends to introduce to **asset providers** a common asset representation format that will be persisted later in a repository of his/her choice. Thus, this common format must consider structural features of repositories and asset specification languages.

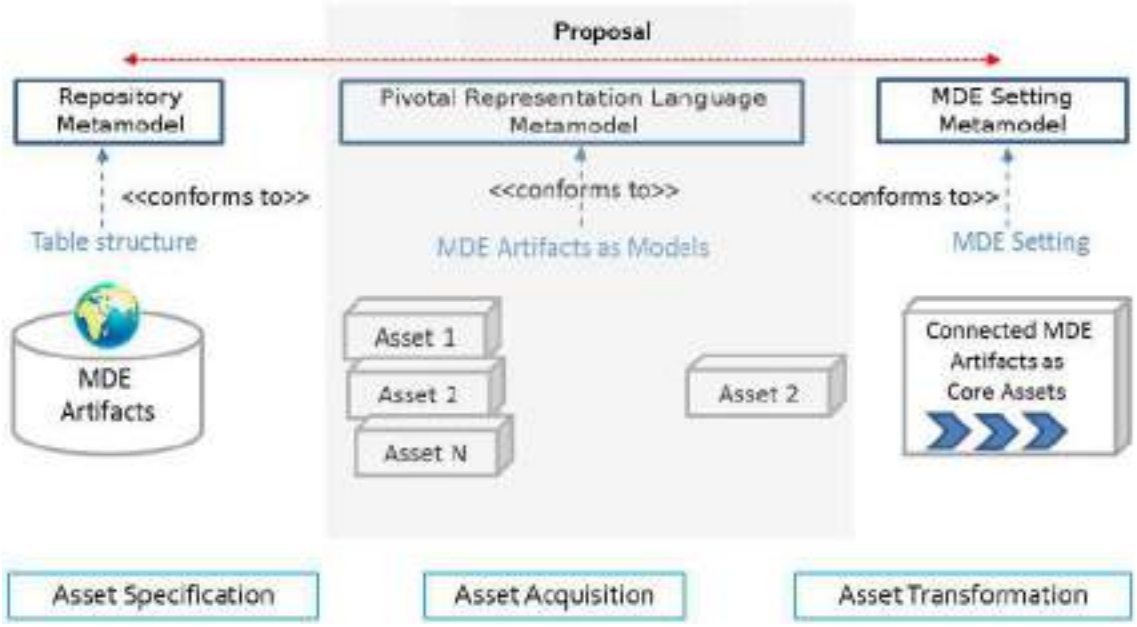


Figure 3.5: Three preliminary phases investigated in this thesis.

Phase 2 - Asset Acquisition intends to provide the means of representation for the **asset consumers** to compare the information of more than one asset. Thus, this common format should support elements for decision-making.

Phase 3 - Asset Transformation intends to provide the means of representation for the **"asset integrator"** to integrate the asset content (MDE Artifacts and Settings) into a tool chain. Thus, the integrator should be able to automatically transform the asset content into a target representation language for tool chain, such as FOMDA DSL, TIL (BIEHL *et al.*, 2014) and etc.

3.4.1 Goal

As illustrates Figure 3.5, the specification phase requires properties found in repository structures, while the Transformation is related with technicalities used with integration toolboxes. In this sense, this section aims at answer the following research question: **Q4: Which studies should we select for evaluation of a common representation language in the context of MDE as a Service?**

3.4.2 Research Method

In order to find out common properties for a pivotal representation language, we conducted six structured reviews of type mapping studies, as shown in Table 3.2. In this sense, suppose that this common format we are looking for is defined by "P", MDE Artifacts are defined by structural features found in asset representations and structural features for tool chain define some integration language for MDE

Artifacts and Settings. Our goal in proposing RAS++ is to connect MDE Artifacts stored in any of these structural features with a common format "P". Latter, "P" is transformed for a representation adopted in a tool chain or asset specification.

Properties introduced in RAS++ are results of an extensive and systematic analysis performed for each phase illustrated in Figure 3.5. Six contributions shown in Table 3.2 are mapping studies by which we extracted properties used in the RAS++ metamodel: **M01** characterizes this research topic with challenges extracted from the literature for implementation of MDE as a Service; **M02** analyzes semantic properties and structural features found in asset representations for software components; **M03** reports some structural features from asset platforms, classifying their intents and properties in support for coopetition; **M04** analyzes existing contributions that support reuse mechanisms for tool chain built on MDE Settings concepts; **M05** provides a list of different types of MDE Toolboxes, and; **M06** intersects previous mappings with a grouping study (RUNESON and HÖST, 2008).

Table 3.2: Mapping studies for opportunities in MDE as a Service

Id	Title	Available at
M01	Characterizing the "MDE as Service" Research Agenda	Section 3.2
M02	Semantic Properties of Software Components	prisma.cos.ufrj.br/wct/ms02.pdf
M03	Intents from Asset Platforms and Their Properties	prisma.cos.ufrj.br/wct/ms03.pdf
M04	MDE Settings Intents and Their Properties	prisma.cos.ufrj.br/wct/ms04.pdf
M05	Diversity of MDE Toolboxes and Their Uncommon Properties	prisma.cos.ufrj.br/wct/ms05.pdf
M06	A Criteria for Representation of Technicalities from MDE Settings and Toolboxes	prisma.cos.ufrj.br/wct/ms06.pdf

Id	Review Type	Research Protocol	Sources
M01	Structured Mapping Study	Snowballing	Researchgate
M02	Ad-hoc Mapping Study	Key-wording	ACM, Researchgate
M03	Ad-hoc Mapping Study	Multi-vocal	Multiple voices
M04	Structured Mapping Study	Key-wording	Scopus, Researchgate
M05	Structured Mapping Study	Snowballing	Scopus, Researchgate
M06	Structured Mapping Study	Coding	Previous mappings

3.4.3 Analysis

In our analysis of data from ad-hoc literature review, we considered 20 representations, as shown in Table 3.3. These publications are classified in tool chain contexts, as shown in the right-side of Figure 3.5, thus presenting some structural feature supporting integration. Table 3.4 shows 20 additional references published between 2013 and march 2015, also classified in tool chain context.

In the context of assets, Table 3.5 presents 29 references. These papers proposed a sort of structural features for classification of artifacts in repositories, as illustrates the left-part of Figure 3.5.

Our analysis concludes that the 69 selected studies represent a suitable population for the execution of our studies, i.e., for evaluation of RAS++ in terms of representativeness. The suitability of this population considers different intents and properties found in 40 representation of structural features for tool chain instantiation, extracted from studies `M04` to `M06`, as well as 29 types of asset representations

Table 3.3: Selected studies for tool chain published until 2012

Id	Paper	Year
S01	G. van Boas. From the workfloor: Developing workflow for the generative model transformer. In 2nd OOPSLA Workshop on Generative Techniques in the context of Model Driven Architecture, 2005.	2005
S02	B. Tekinerdogan, S. Bilir, and C. Abatlevi. Integrating platform selection rules in the model driven architecture approach. In Proceedings of the 2003 European conference on Model Driven Architecture: foundations and Applications, MDAFA'03, pages 159-173, 2005.	2005
S03	J. P. Almeida, R. Dijkman, M. Sinderen, and L. F. Pires. Platform-independent modelling in mda: Supporting abstract platforms. In Model Driven Architecture Foundations and Applications, MDAFA'05, pages 174-188, 2005.	2005
S04	D. Wagelaar. Blackbox composition of model transformations using domain-specific modelling languages. In Proceedings of the ECMDA Composition of Model Transformations Workshop, pages 15-19, 2006.	2006
S05	B. Vanhooff, D. Ayed, and Y. Berbers. A framework for transformation chain development processes. In Proceedings of the ECMDA Composition of Model Transformations Workshop, pages 3-8, 2006.	2006
S06	B. Vanhooff, S. V. Baelen, A. Hovsepyan, W. Joosen, and Y. Berbers. Towards a transformation chain modeling language. In Proceedings of the 6th international conference on Embedded Computer Systems: architectures, Modeling, and Simulation, SAMOS'06, pages 39-48, 2006.	2006
S07	F. P. Basso, L. B. Becker, and T. C. Oliveira. Using the fomda approach to support object-oriented real-time systems development. In Object and Component-Oriented Real-Time Distributed Computing, 2006. ISORC 2006. pages 374-381, 2006.	2006
S08	M. Völter and I. Groher. Handling variability in model transformations and generators. In Proceedings of the 7th OOPSLA Workshop on Domain-Specific Modeling, DSM'07, 2007.	2007
S09	J. Steel, J.M. Jézéquel, On model typing, Software & Systems Modeling 6 (4) (2007) 401-413, ISSN 1619-1366.	2007
S10	R. Bendraou, P. Desfray, M. Gervais, A. Muller MDA Tool Components: a proposal for packaging know-how in model driven development .Software & Systems Modeling, 7, 3, 329-343, 2009.	2009
S11	S. Trujillo, A. Zubizarreta, X. Mendialdua, and J. de Sosa. Feature-oriented refinement of models, metamodels and model transformations. In Proceedings of the First International Workshop on Feature-Oriented Software Development, FOSD'09, pages 87-94, 2009.	2009
S12	J. M. Kuster, T. Gschwind, and O. Zimmermann. Incremental development of model transformation chains using auto mated testing. In Model Driven Engineering Languages and Systems, MODELS'09, pages 733-747, 2009.	2009
S13	A. Etien, A. Muller, T. Legrand, and X. Blanc. Combining independent model transformations. In Proceedings of the 2010 ACM Symposium on Applied Computing, SAC'10, pages 2237-2243, 2010.	2010
S14	M. Azanza, D. Batory, O. Diaz, and S. Trujillo. Domain-specific composition of model deltas. In Theory and Practice of Model Transformations, volume 6142 of Lecture Notes in Computer Science, pages 16-30, 2010.	2010
S15	J. S. Cuadrado, E. Guerra, and J. Lara. Generic model transformations: Write once, reuse everywhere. In Theory and Practice of Model Transformations. 6707:62-77, 2011.	2011
S16	M. Asztalos, E. Syriani, M. Wimmer, and M. Kessentini. Simplifying model transformation chains by rule composition. In J. Dingel and A. Solberg, editors, Models in Software Engineering, volume 6627 of Lecture Notes in Computer Science, pages 293-307. Springer Berlin Heidelberg, 2011.	2011
S17	A. Yie, R. Casallas, D. Deridder, and D. Wagelaar. Realizing model transformation chain interoperability. Software & Systems Modeling, 11(1):55-75, 2012.	2012
S18	P. Quyet-Thang and A. Beugnard. Automatic adaptation of transformations based on type graph with multiplicity. In Software Engineering and Advanced Applications (SEAA), 2012 38th EUROMICRO Conference on, pages 170-174, 2012.	2012
S19	A. Etien, V. Aranega, X. Blanc, and R. F. Paige. Chaining model transformations. In Proceedings of the First Workshop on the Analysis of Model Transformations, AMT'12, pages 9-14, 2012.	2012
S20	V. Aranega, A. Etien, and S. Mosser. Using feature models to tame the complexity of model transformation engineering. In ACM/IEEE 15th International Conference on Model Driven Engineering Languages and Systems MODELS 2012, 2012.	2012

proposing different structural features extracted from studies `M01` to `M03`. Moreover, it should be noted that, through them, we extracted structural features for a common representation language: asset representations found in studies shown in Table 3.5 and tool chain representations found in studies shown in Tables 3.3 and 3.4.

3.4.4 Final Remarks

These 69 contributions from the aforementioned tables present structural features that can be used in one or another context of a scenario for coopetition in MDE as a Service. This means that one contribution can represent well for part, but not all, the illustrated scenario in Figure 3.5. Due to the lack of a common/pivotal representation language, such contexts are not properly connected in existing contributions for representations. In a coopetition scenario, this will imply in a manual effort for integration, which is costly (MOHAGHEGHI *et al.*, 2013).

Table 3.4: Selected studies for tool chain published after 2012

Id	Paper	Year
S21	R. Hebig, H. Giese, F. Stallmann, A. Seibel, On the Complex Nature of MDE Evolution. Proceedings of Model Driven Engineering Languages and Systems, MODELS 2013, 436-453, 2013.	2013
S22	A. Vignaga, F. Jouault, M. C. Bastarrica, H. Bruneliere, Typing artifacts in megamodeling. Software & Systems Modeling 12 (1) (2013) 105-119, ISSN 1619-1366.	2013
S23	E. Guerra, J. de Lara, D. S. Kolovos, R. F. Paige, O. M. dos Santos, Engineering model transformations with transML. Software & System Modeling 12 (3) (2013) 555-577.	2013
S24	F. P. Basso, R. M. Pillat, T. C. Oliveira, L. B. Becker, Supporting Large Scale Model Transformation Reuse. International Conference on Generative Programming: Concepts & Experiences. GPCE'13, 169-178, 2013.	2013
S25	C. Alvarez, R. Casallas, MTC Flow: A Tool to Design, Develop and Deploy Model Transformation Chains. Workshop on ACadeMics Tooling with Eclipse, ACME'13, ISBN 978-1-4503-2036-8, 7:1-7:9, 2013.	2013
S26	A. Etien, A. Muller, T. Legrand, R. F. Paige, Localized model transformations for building large-scale transformations. Software & Systems Modeling (2013) 1-25 ISSN 1619-1366.	2013
S27	F. P. Basso, C. M. L. Werner, R. M. Pillat, T. C. Oliveira, A Common Representation for Reuse Assistants. 13th International Conference on Software Reuse, ICSR'13, 283-288, 2013.	2013
S28	A. F. M. Mascarenhas, A. Andrade, R. P. Maciel, MTP: Model Transformation Profile. Software Components, Architectures and Reuse (SBCARS), 109-118, 2013.	2013
S29	L. Rose, E. Guerra, J. Lara, A. Etien, D. Kolovos, R. Kolovos, Genericity for model management operations. Software & Systems Modeling 12 (1) (2013) 201-219.	2013
S30	V. O. Costa, J. M. B. O. Junior, L. G. P. Murta, Semantic Conflicts Detection in Model-driven Engineering. International Conference on Software Engineering and Knowledge, 2013., 2145 656-661, 2013.	2013
S31	J. S. Cuadrado, E. Guerra, J. D. Lara, A Component Model for Model Transformations. IEEE Transactions on Software Engineering 40 (11) (2014) 1042-1060.	2014
S32	J. S. Cuadrado, E. Guerra, J. D. Lara, M. Biehl, J. El-Khoury, F. Loiret, M. Torngren, On the modeling and generation of service-oriented tool chains. Software & Systems Modeling 13 (2) (2014) 461-480, ISSN 1619-1366.	2014
S33	F. P. Basso, C. M. L.Werner, T. C. Oliveira. Towards Facilities to Introduce Solutions for MDE in Development Environments with Reusable Assets. International Conference on Information Reuse and Integration, IRI'14, 195-202, 2014.	2014
S34	J. Vara, V. Bollati, A. Jimenez, E. Marcos, Dealing with Traceability in the MDD of Model Transformations. Transactions on Software Engineering 40 (6) (2014) 555-583.	2014
S35	K. Garces, J. M. Vara, F. Jouault, E. Marcos, Adapting transformations to metamodel changes via external transformation composition. Software & Systems Modeling 13 (2) (2014) 789-806, ISSN 1619-1366.	2014
S36	F. P. Basso, T. C. Oliveira, K. Farias, Extending JUnit 4 with Java Annotations and Reflection to Test Variant Model Transformation Assets. Symposium On Applied Computing, SAC'14, 1601-1608, 2014.	2014
S37	F. P. Basso, R. M. Pillat, T. C. Oliveira, M. D. D. Fabro, Generative Adaptation of Model Transformation Assets: Experiences, Lessons and Drawbacks. Symposium On Applied Computing, SAC'14, 1027-1034, 2014.	2014
S38	L. A. Rahim, J. Whittle, A survey of approaches for verifying model transformations. Software & System Modeling 14 (2) (2015) 1003-1028.	2015
S39	E. Syriani, H. Vangheluwe, B. LaShomb, T-Core: a framework for custom-built model transformation engines. Software & Systems Modeling 14 (3) (2015) 1215-1243.	2015
S40	J. S. Cuadrado, E. Guerra, J. de Lara, Reusable Model Transformation Components with bentō. International Conference on Theory and Practice of Model Transformations, ICMT 2015, 59-65, 2015.	2015

Table 3.5: Selected studies with structural features for assets

Id	Paper	Year
S41	RAS Reusable Asset Specification Version 2.2 November 2005. Av. at <http://www.omg.org/spec/RAS/>.	2005
S42	G. ELIAS, M. SCHUENCK, Y. NEGÓCIO, X-ARM: an asset representation model for component repository systems. Proceedings of the 2006 ACM symposium on Applied computing, SAC'06, pp. 1690-1694.	2006
S43	S. PARK, S. PARK, V. SUGUMARAN, Extending reusable asset specification to improve software reuse. Proceedings of the 2007 ACM symposium on Applied computing, SAC'07, pp. 1473-1478.	2007
S44	R. FRANCE, J. BIEMAN, B. CHENG, 2007, Repository for Model Driven Development (ReMoDD). Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 4364 LNCS, pp. 311-317.	2007
S45	H. B. HADJI, K. SU-KYOUNG, C. HO-JIN, A Representation Model for Reusable Assets to Support User Context. IEEE International Symposium on Service-Oriented System Engineering, SOSE'08, pp. 91-96.	2008
S46	I. ELGEDAWY, 2009, Reusable SOA Assets Identification Using E-Business Patterns. World Conference on Services-II, 2009. SERVICES-2'09. pp. 33-40.	2009
S47	R. Hong-min, Y. Zhi-ying, Z. Jing-zhou, Design and Implementation of RAS-Based Open Source Software Repository. Sixth International Conference on Fuzzy Systems and Knowledge Discovery, 2009. FSKD'09., vol. 2, 219-223, 2009.	2009
S48	Asset Management Specification. Av. at <http://open-services.net/wiki/asset-management/>	2009
S49	R. P. DOS SANTOS, C. WERNER, 2010, Analyzing the Concept of Components in the Brechó-VCM Approach through a Sociotechnical and Software Reuse Management Perspective. Software Components, Architectures and Reuse (SBCARS), 2010 Fourth Brazilian Symposium on, pp. 21-30.	2010
S50	M. Wimmer, S. Martinez, F. Jouault, J. Cabot, A Catalogue of Refactorings for Model-to-Model Transformations. Journal of Object Technology 11 (2) (2012) 2:1-40, ISSN 1660-1769.	2012
S51	W. Zhang, B. Moller-Pedersen, M. Biehl, A Light-weight Tool Integration Approach - From a Tool Integration Model to OSLC Integration Services. International Conference on Software Engineering and Applications, ICSOFT'12, 137-146, 2012.	2012
S52	Z. Weiqing, V. Leilde, B. Moller-Pedersen, J. Champeau, C. Guychard, Towards Tool Integration through Artifacts and Roles. Software Engineering Conference (APSEC), 2012 19th Asia-Pacific, 603-613, 2012.	2012
S53	P. V. Gorp, Paul W. P. J. Grefen, Supporting the internet-based evaluation of research software with cloud infrastructure. Software and System Modeling. (11), 1, pp 11-28. 2012	2012
S54	A. Vignaga, F. Jouault, M. C. Bastarrica, H. Bruneliere, Typing artifacts in megamodeling. Software & Systems Modeling 12 (1) (2013) 105-119, ISSN 1619-1366.	2013
S55	N. Tran, A. Ganser, H. Lichter, Multi Back-Ends for a Model Library Abstraction Layer. Computational Science and Its Applications, vol. 7973 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 160-174, 2013.	2013
S56	W. Zhang, B. Moller-Pedersen, Establishing tool chains above the service cloud with integration models. IEEE 20th International Conference on Web Services, ICWS 2040 2013, 372-379, 2013.	2013
S57	M. Elaasar, A. Neal, Integrating Modeling Tools in the Development Lifecycle with OSLC: A Case Study. Model Driven Engineering Languages and Systems, MODELS'13, 154-169, 2013.	2013
S58	F. P. Basso, C. M. L. Werner, R. M. Pillat, T. C. Oliveira, A Common Representation for Reuse Assistants. 13th International Conference on Software Reuse, ICSR'13, 283-288, 2013.	2013
S59	R. P. DOS SANTOS, M. G. P. ESTEVES, G. DE S. FREITAS, et al., Software Ecosystems Comprehension and Evolution. Social Networking, v. 3, n. 2 (Feb), pp. 108-118. 2013	2013
S60	B. Combemale, J. Deantoni, B. Baudry, R. France, J.M. Jézéquel, J. Gray, Globalizing Modeling Languages. IEEE Computer, Institute of Electrical and Electronics Engineer 47 (6) (2014) 68-71.	2014
S61	L. Lucio, M. Amrani, J. Dingel, L. Lambers, R. Salay, G. M. Selim, E. Syriani, M. Wimmer, Model transformation intents and their properties. Software & Systems Modeling (2014) 1-38, ISSN 1619-1366.	2014
S62	F. Basciani, D. D. Ruscio, L. Iovino, A. Pierantonio, Automated Chaining of Model Transformations with Incompatible Metamodels. Model-Driven Engineering Languages and Systems, 602-618, 2014.	2014
S63	F. Basciani, J. D. Rocco, D. D. Ruscio, A. D. Salle, L. Iovino, A. Pierantonio, MDEForge: An extensible Web-based modeling platform. Workshop on Model-Driven Engineering on and for the Cloud, CloudMDE 2014, 66-75, 2014.	2014
S64	W. Zhang, B. Moller-Pedersen, Modeling of tool integration resources with OSLC support. Model-Driven Engineering and Software Development, MODELSWARD, 99-110, 2014	2014
S65	J. D. Rocco, D. D. Ruscio, L. Iovino, A. Pierantonio, Collaborative Repositories in Model-Driven Engineering. IEEE Software 32 (3) (2015) 28-34.	2015
S66	J. Criado, S. Martinez, L. Iribarne, J. Cabot, Enabling the reuse of stored model transformations through annotations. International Conference on Model Transformations, 1-15, 2015.	2015
S67	W. Zhang, Tool Integration by Models, Not Only by Metamodels - Applying Modeling to Tool Integration. Model-Driven Engineering and Software Development, MODELSWARD, 461-469, 2015.	2015
S68	D. BADAMPUDI, C. WOHLIN, K. PETERSEN, Software component decision-making: In-house, OSS, COTS or outsourcing - A systematic literature review. Journal of Systems and Software, v. 121, pp. 105 - 124. 2016	2015
S69	T. LIMA, R. P. DOS SANTOS, J. OLIVEIRA, et al., The importance of socio-technical resources for software ecosystems management. Journal of Innovation in Digital Ecosystems, v. 3, n. 2, pp. 98 - 113. 2016	2015

Chapter 4

RAS++

> Everything must be made as simple as possible. But not simpler.
>
> Albert Einstein

As result from a process that extracted structural features from analytical studies and literature mappings discussed in previous chapters, this chapter presents a new asset specification language called RAS++. This way, in order to reach common properties, we investigate an open question: **Q5: Which are the properties required in RAS++ in the context of MDE as a Service for asset representation?**

Next, we depict common properties found in three preliminary phases for tool chain: *Specification* is presented in Section 4.1, Section 4.2 represents assets for *Acquisition* and assets for *Transformation* are depicted in Section 4.3.

4.1 Asset Specification

The need for coopetition in MDE is recent (COMBEMALE *et al.*, 2014; MUSSBACHER *et al.*, 2014). Authors claim that a repository is necessary for asset consumers to easily find and comprehend what is necessary to introduce artifacts in practice. However, not much is understood about the requirements for the implementation of coopetition in MDE. For this reason, COMBEMALE *et al.* (2014) claimed that currently they have more questions than answers in this matter.

As illustrated in Figure 4.1, our goal is to build a representation language to be used together with Knowledge Bases (KB)/repositories (COMBEMALE *et al.*, 2014; FRANCE *et al.*, 2007; GORP and MAZANEK, 2011; JACOBSON *et al.*, 2012; ROCCO *et al.*, 2016). Clearly, these proposals present possibilities for implementation of coopetition in MDE as a Service, but they are unconnected. A common representation language for MDE could connect clients with many platforms in support

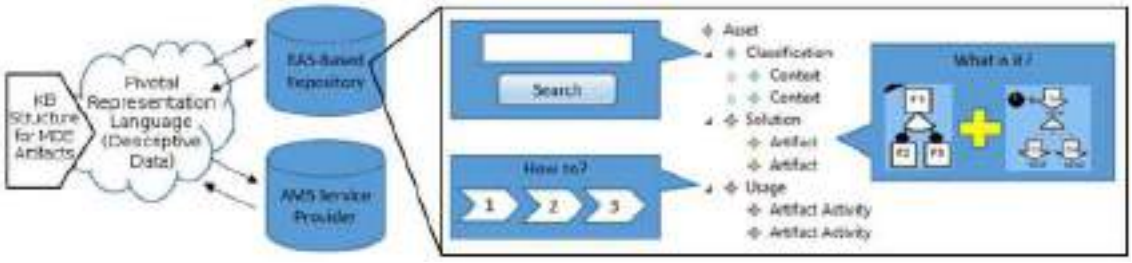


Figure 4.1: The role of assets to a KB for MDE Artifacts.

for MDE Artifacts, but we miss this language and its requirements. In definition, the closer in the state of the art from the conception of this type of language are two asset specifications: Reusable Asset Specification (RAS) (OMG, 2005) and Asset Management Specification (AMS) (AMS, 2014). Asset specifications are acknowledged for years to play the role of a pivot between clients and repositories. In this sense, we investigate in this chapter the requirements from assets specifications to support the *Specification* phase through an analytical study.

The presentation of this phase is organized as follows. Section 4.1.1 exposes the analytical study with demonstrations presented in Section 4.1.2. This analysis provided some limitations used for the conception of RAS++, which is depicted in Section 4.1.3. Finally, conclusions are discussed in Section 4.1.4.

4.1.1 Mapping Study

In the following, we describe the conception of RAS++ in support for the *Specification* phase. So far, the acknowledged role of a pivotal representation language is to allow automatically publish and download the information independently from a database structure adopted to store the artifacts. It also must represent descriptive information associated with artifacts in a structured and common format. However, the conception of a new representation language for coopetition in MDE as a Service is far more complex and need specific analysis about the properties from these specifications. For example, due to possibility of wrong constructions for MDE Artifacts and Settings, RAS++ must avoid ambiguities, be representative and in the same way not allow wrong representations.

Goal

In order to answer the more general research question Q5, our study is subdivided in five sub-questions. This section answers the first one: **Q5.1) Which properties are required in RAS++ to support coopetition in the Specification phase?**

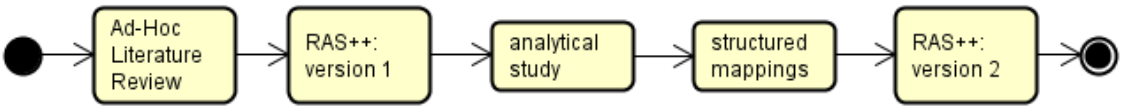


Figure 4.2: Research method to derive common properties in RAS++.

Research Method

This question needs the execution of an analysis of these repositories and asset specifications. However, to the best of our knowledge, there is no research method instructing how to derive a common representation language. For this reason, our research approach is quite ad-hoc in this chapter, but it includes some empirical steps shown in Figure 4.2.

In order to answer **Q5.1**, we first conducted an ad-hoc literature review about assets along three software reuse disciplines. We concluded that RAS and AMS are the standard definitions for assets and are on the top of repositories, selecting them as basis for construction of RAS++. The second step is the conception of RAS++ in the first version of DSL, which was tested with a proof of concept

We then conducted an analytical study, comparing RAS and AMS (BASSO *et al.*, 2016a) through a case study on the representation of some toolboxes for Software Product Lines (SPL) (THÜM *et al.*, 2014), found in the literature of the area, as asset models. These toolboxes are used in practice, achieving a mature level of adoption in some organizations (BERGER *et al.*, 2013; CAPILLA *et al.*, 2014). Besides, SPL toolboxes (BERGER *et al.*, 2013) and Dynamic SPL toolboxes (CAPILLA *et al.*, 2014) are considered as MDE Artifacts, too (FRANCE *et al.*, 2006).

Property Selection Criteria

Since RAS++ is a result of a long process of analysis of properties for assets found in repositories for software components, MDE repositories and asset specification languages, a criteria for inclusion of metaclasses and properties is needed. We adopted the following criteria in the final version of RAS++:

1. The more representative metaclass is used in RAS++;

2. In case of a tie, the following priority is used: metaclasses from UML (used as basis to support light-weight extensibility), metaclasses from RAS and metaclasses from AMS;

3. In case of conceptual ambiguities for representation of the same data using different metaclasses, we selected the metaclass that better describes the concepts;

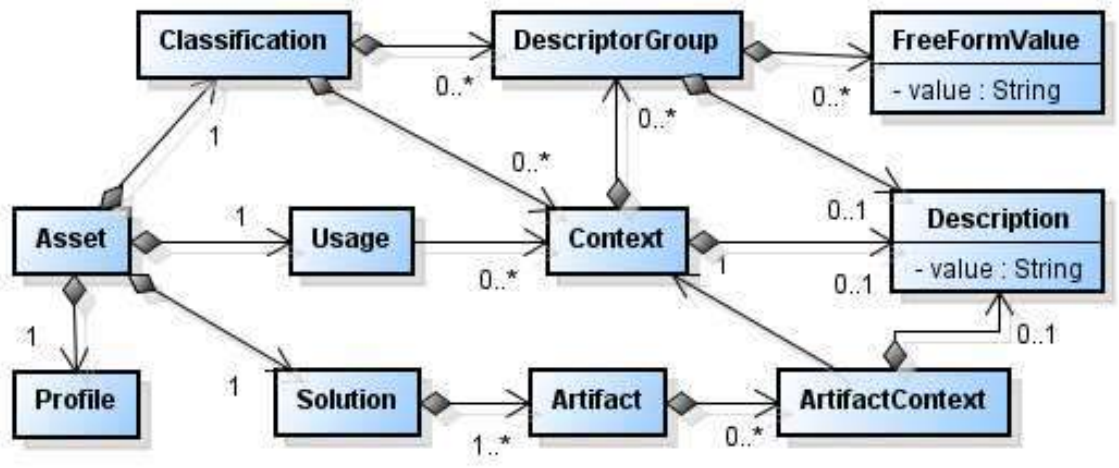


Figure 4.3: Main metaclasses from RAS to represent descriptive data.

4. In case of hierarchical ambiguities, i.e., equivalent properties found in class hierarchies of RAS++, the property of sub-classes is removed, and;

5. To keep the originality and focus in our contribution, we did not include other extensions proposed for RAS or AMS in RAS++, thus removing them (including the RAS Component Profile) from the first RAS++ version.

We also revisited and refined metaclasses and properties introduced in RAS++ along four years. For example, some ambiguities have been found when introducing properties in RAS++ in support for the *Transformation* phase, so we revisited the *Specification* phase to remove them. In order to determine which metalass is more representative for the *Specification* phase, we considered the following descriptive data for MDE Artifacts: 1) catalog information used for searching DSLs based on standard data; 2) instructive information about how to use and adapt existing components and; 3) descriptive information for one to decide the best option for DSLs, that best meets a specific need in a software development context.

4.1.2 Asset Specification Languages

Reusable Asset Specification

RAS is an OMG standard to classify, catalog, and instruct the reuse of software artifacts in reuse repositories (OMG, 2005). RAS provides meta-classes shown in Figure 4.3 to detail instructive information associated with artifacts, used by end-users to learn about what should be adapted in existing software artifacts for different needs in software projects (ELGEDAWY, 2009).

Figure 4.1 exemplifies an asset for a scenario where RAS specifications are currently used: *to describe information on how to reuse some software component.*

The asset exemplifies some information used to detail artifacts that compose two domain models used by a model transformation engine to produce information systems through model transformations. These domain models are used to generate code for many information systems. Moreover, the content of the asset (the element *Solution*) is the target of a technical solution for MDE, which adapts the artifacts for specific target platforms (e.g., to generate code for different Java APIs) (BASSO *et al.*, 2013a).

In order to promote the reuse of these two software artifacts in future software development projects, it is necessary to provide adequate information used in specific moments in a reuse process (KRUEGER, 1992; OMG, 2005). This way, Figure 4.1 shows that an asset structures descriptive information that can be used at different moments in between acquisitions from repositories to the integration of the artifacts in target software projects. 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 for end-users to learn (see *Usage*) how to adapt and integrate the asset content (see the artifacts in the *Solution* structure) in a target software project. Thus, this structure is important to localize the correct information according to reuse steps (KRUEGER, 1992).

Asset Management Specification

Another option to specify an asset is AMS, an industrial specification (AMS, 2014) to catalog tools that help in software engineering tasks (ELAASAR and NEAL, 2013). AMS is part of Open Services for Lifecycle Collaboration (OSLC) (OSLC, 2017a), an industry specification that allows to chain tools through asset specifications and web services. In other words, tools that have integration interfaces used for execution support (starting points) specified in OSLC are integrated (ELAASAR and NEAL, 2013). OSLC facilitates the chaining of software engineering tasks assisted by tools, since interfaces are specified in a common format. Substantially, for the problem motivated in this work, AMS allows to represent descriptive information in a standard format that follows the metamodel shown in Figure 4.4.

Conceptual Demonstration

In the following, we summarized our comparison of RAS and AMS (BASSO *et al.*, 2016a). Currently, end-users have no access to information about Toolboxes for SPL without performing an extensive study in papers, web pages and tutorials. A negative point is that there is no summarized information that can be crossed to make decisions.

Catalog and Description

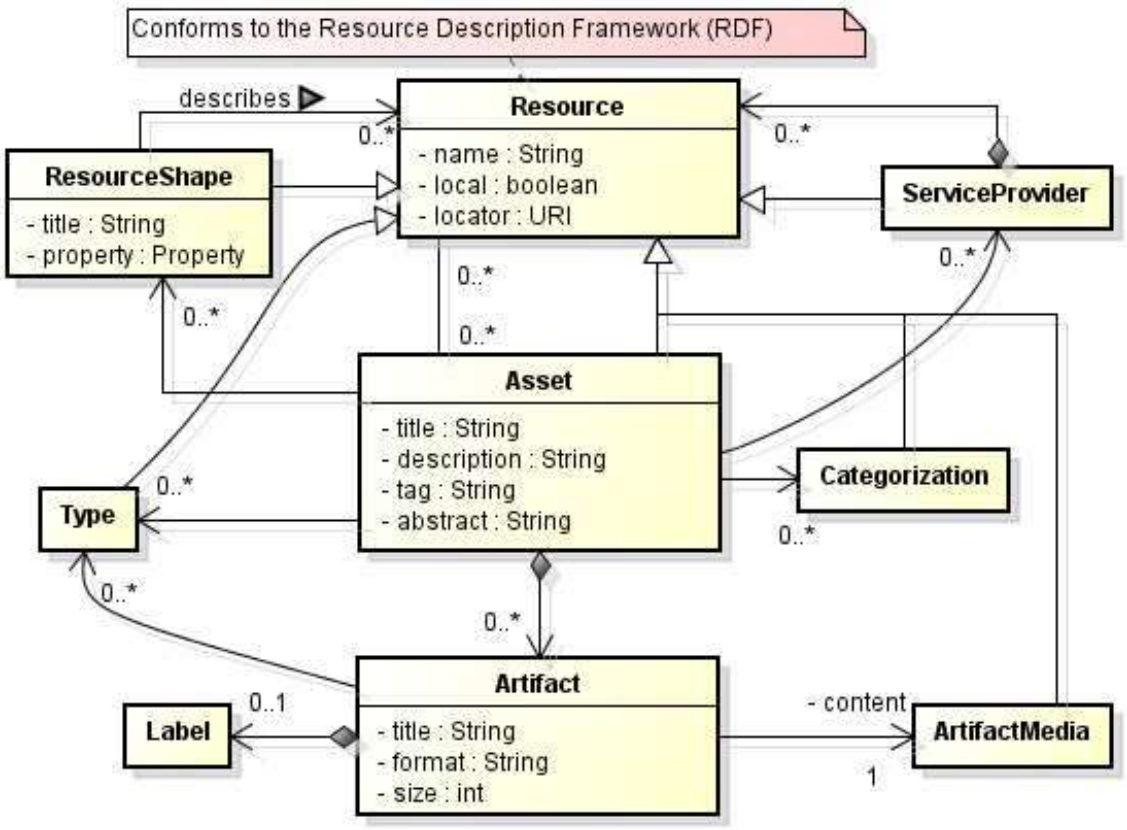


Figure 4.4: Metaclasses from AMS

RAS and AMS provide an element for classification, which can receive information about many contexts in which the asset is applied. This information is exemplified in the asset shown in Figure 4.5 (A), represented in AMS. Figure 4.5 (B) shows an asset specified with the standard RAS. Assets describe the FOMDA DSL, a solution to design and adapt model transformation chains, exemplified as the artifacts in Figure 4.1.

AMS is restricted for keyword search. However, more than keyword-based search is needed. Hence, the example shown in Figure 4.1 adds two new elements: descriptor groups and free form values, only available in RAS. Descriptors groups are used to add structured textual information to better describe a context associated with an asset.

Both specifications provide a mean to contextualize technical solutions for MDE, i.e., to provide a classification and description. However, only in RAS it is possible to represent rich and structured textual content. Examples of other works that increment descriptions about assets are found in (ELGEDAWY, 2009; HADJI *et al.*, 2008; HONG-MIN *et al.*, 2009; PARK *et al.*, 2007), which extend these meta-data to add standard taxonomies to classify specific types of software components. Thus, it is possible to explore these structures to provide taxonomies to better classify and catalog technical solutions for MDE.

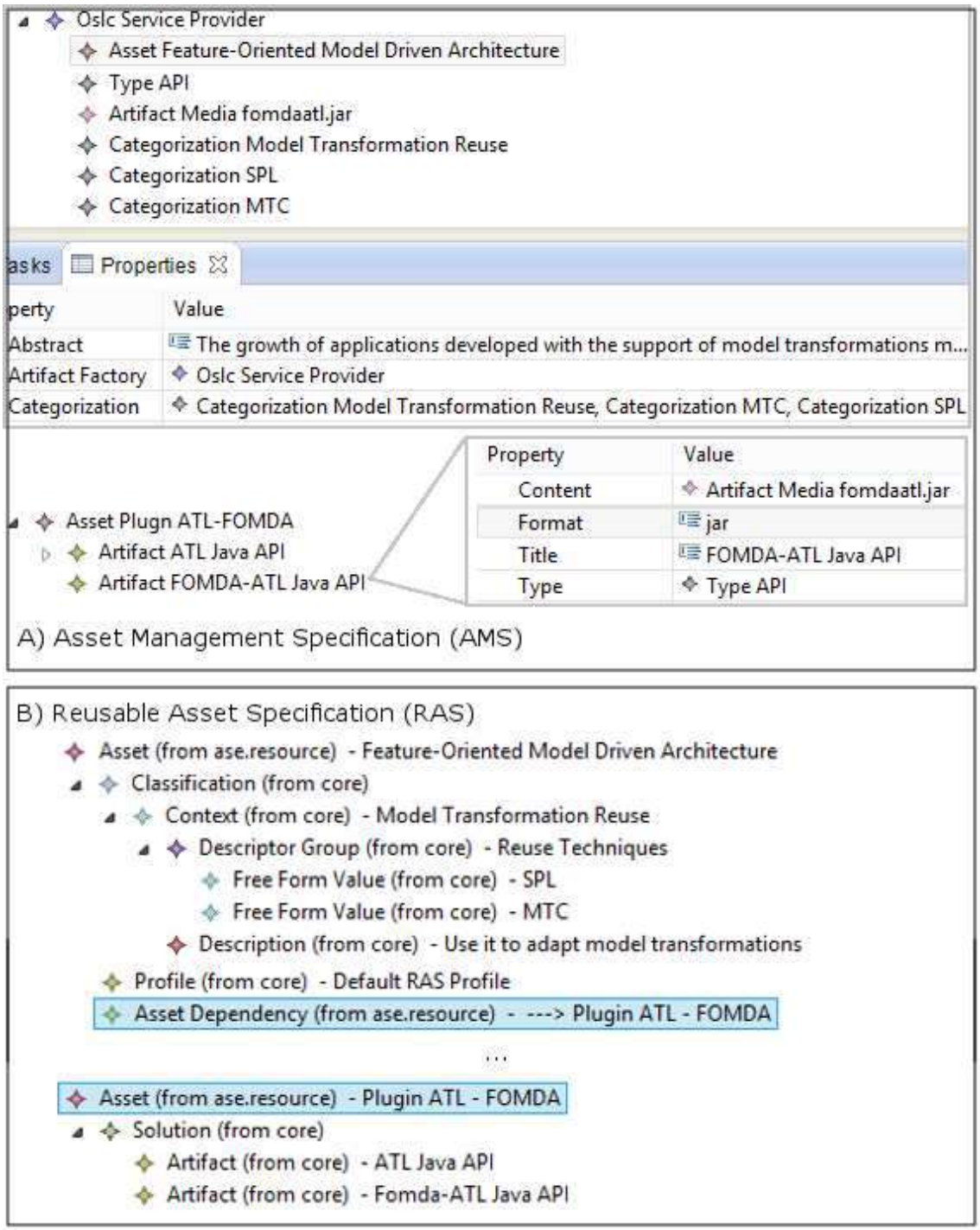


Figure 4.5: Classification of the same toolbox with AMS and RAS.

Artifacts

Artifacts represented with RAS and AMS are mere links to physical artifacts, called as artifact media resource in OSLC. In addition, RAS also allows packing artifacts in compressed files and AMS allows retrieving artifacts through calls for web services. In a coopetition scenario for MDE as a Service, we considered that artifacts are distributed among several repositories that can be based on RAS (a

lot of information is still represented with RAS) or OSLC (there is a tendency to represent assets with AMS). For our purposes it is important that artifacts can be retrieved through proxies such as OSGI and MAVEN or class for web services. In the case of artifacts described by AMS, this information is specified with an RDF/XML described in OSLC. The RAS meta-class, namely *Artifact*, does not allow the representation of such important information.

Relationships

In many cases, assets must establish relationships. So, it is important to define dependency relationships between data expressed in assets. Figure 4.5 (B) exemplifies this situation in which FOMDA DSL depends on another asset called Plugin ATL-FOMDA. This is an API which makes an integration with a known model transformation engine called ATL (BÉZIVIN, 2005), used to execute model transformation written in ATL language. This engine could be also represented as an asset, which means that, through dependencies, it is possible to establish traces between many assets and MDE Artifacts.

Dependencies among artifacts, among assets and artifacts are also allowed in both specifications. However, we found that dependencies are more expressive in AMS because they provide information about service providers (repositories), while in RAS the dependencies are assumed to be established in assets from the same repository. This is a difference that enables AMS specifications to associate assets from many repositories on the web, while RAS limits the use for "in-company" solutions. Thus, this property is important for implementation of coopetition considering the diversity of repositories in support for MDE Artifacts.

Content Figure 4.1 exemplified the *Solution* structure, the content-part of an asset, which is usually applied to describe software artifacts (e.g., models, components, source code). Artifacts can also represent semantics from MDE Artifacts, as illustrates Figure 4.6. In this case, an asset can have information associated with artifacts such as configuration files, APIs and libraries, binaries and scripts, services and model transformations, among others. Besides, these solutions also provide activities in support for Software Engineering tasks (BATORY *et al.*, 2013b). Therefore, it is important to abstract the data associated with operations and artifacts that a given solution offers for end-users.

The specifications allow to represent artifacts and tasks (this one only in RAS), but with a very limited set of meta-classes.

Instructions

As illustrated in Figure 4.1, in assets that describe reusable artifacts for software components it is common to associate instruction for reuse. However, it is not so obvious what should be represented as instruction in an asset for a technical solution for MDE, such as tools and model transformations. Tools and transformations need

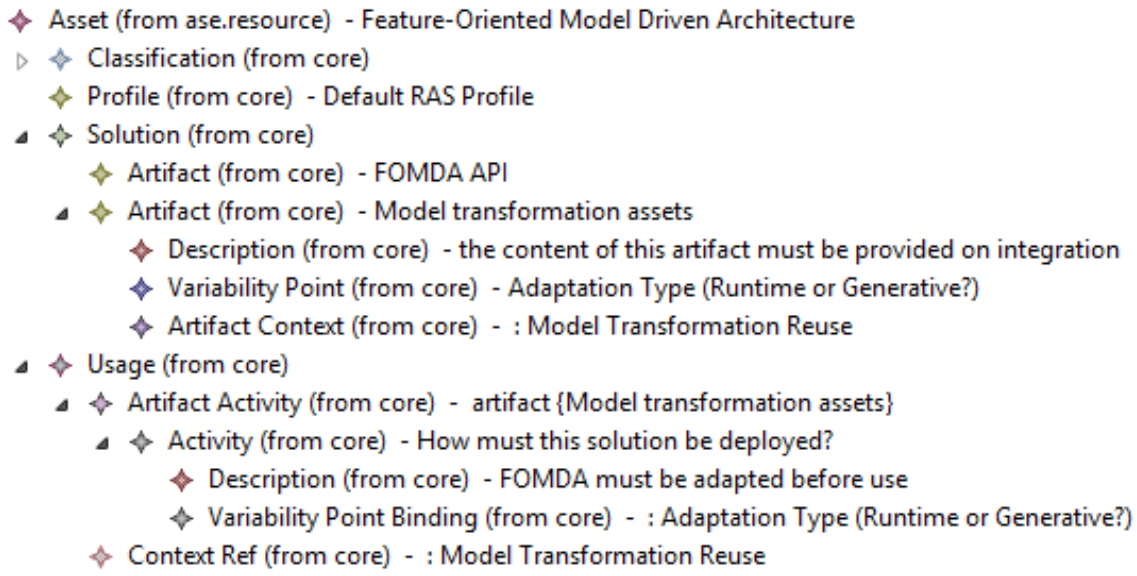


Figure 4.6: Content and instruction associated with the technical solution called FOMDA.

to be adapted (WHITTLE *et al.*, 2013) and they also provide activities in support for Software Engineering tasks (BASSO *et al.*, 2014b). Thus, it is needed to configure a set of libraries, components and files.

Figure 4.6 illustrates some of the elements provided in RAS to support such an instructive information. In order to give to this tool a different way to work, an artifact can receive descriptive information about variability points that highlight in a tool the main features that can be changed. For example, model transformation assets that configure FOMDA DSL in Figure 4.6 can be configured to support runtime or generative adaptations over those artifacts detailed in Figure 4.1. This is a benefit that allows end-users to understand how a tool works and how to change it.

The instruction on how to proceed to apply an adaptation is specified in activities inside the *Usage* structure shown in Figure 4.7, 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, with this information, it is answered how one can use the solution.

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.

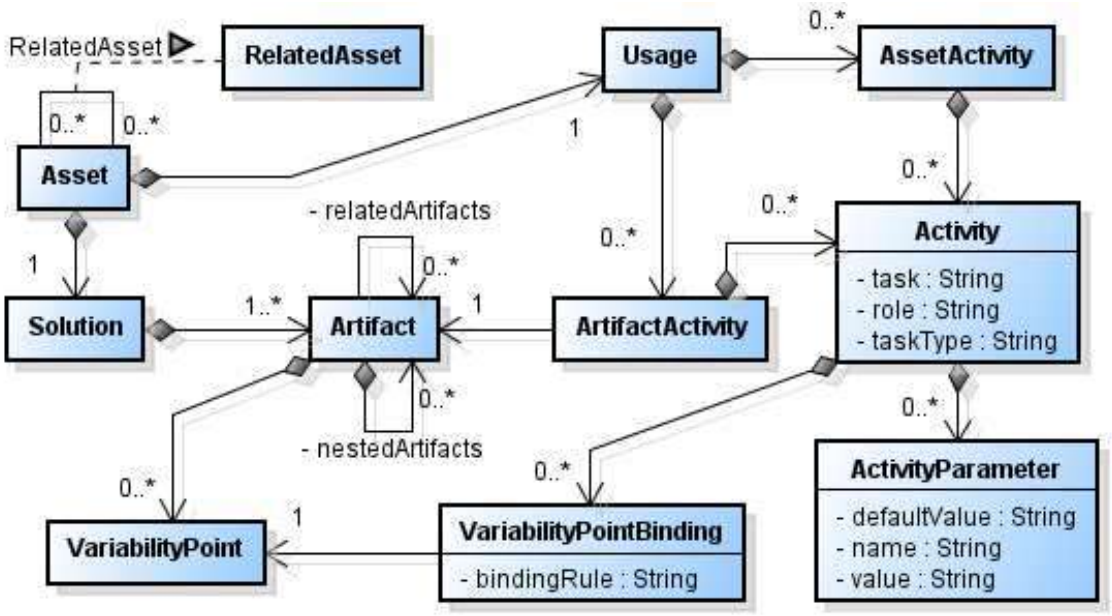


Figure 4.7: Metaclasses from RAS to represent activities.

4.1.3 RAS++ Metamodel

This section answers the research question **Q5.1) Which properties are required in RAS++ to support coopetition in the Specification phase?** We found that, to match these specifications, all their properties are relevant. This requires merging in RAS++ the properties found in RAS and AMS. The RAS++ metamodel, therefore, is built on properties from both RAS and AMS, as illustrated in Figure 4.10.

RAS++ Overview

Figure 4.8 shows structural features from RAS++: In (A) and (B) are metaclasses from RAS in conformity with the meta-levels described in (C). RAS++ merges features from RAS and AMS in EMF models. In the following, some extensions for RAS are presented, thus making RAS++ a super-set of RAS and AMS.

Figure 4.8 (A) and (B) show two metaclasses that are central for an asset representation language: `Artifact` and `Asset`. Associated with assets and artifacts there are many other metaclasses that aim at specifying semantics for classification, description, context, types, etc.

Figure 4.8 (A) shows that artifacts also allow the representation of sub-artifacts, variability points and artifact type, important data to represent MDE Artifacts. In addition, Figure 4.8 (B) shows that assets allow the representation of information for version, state and creation date, thus mixing information generated by a repository for management of assets and information provided by the software engineer for the description of a set of artifacts.

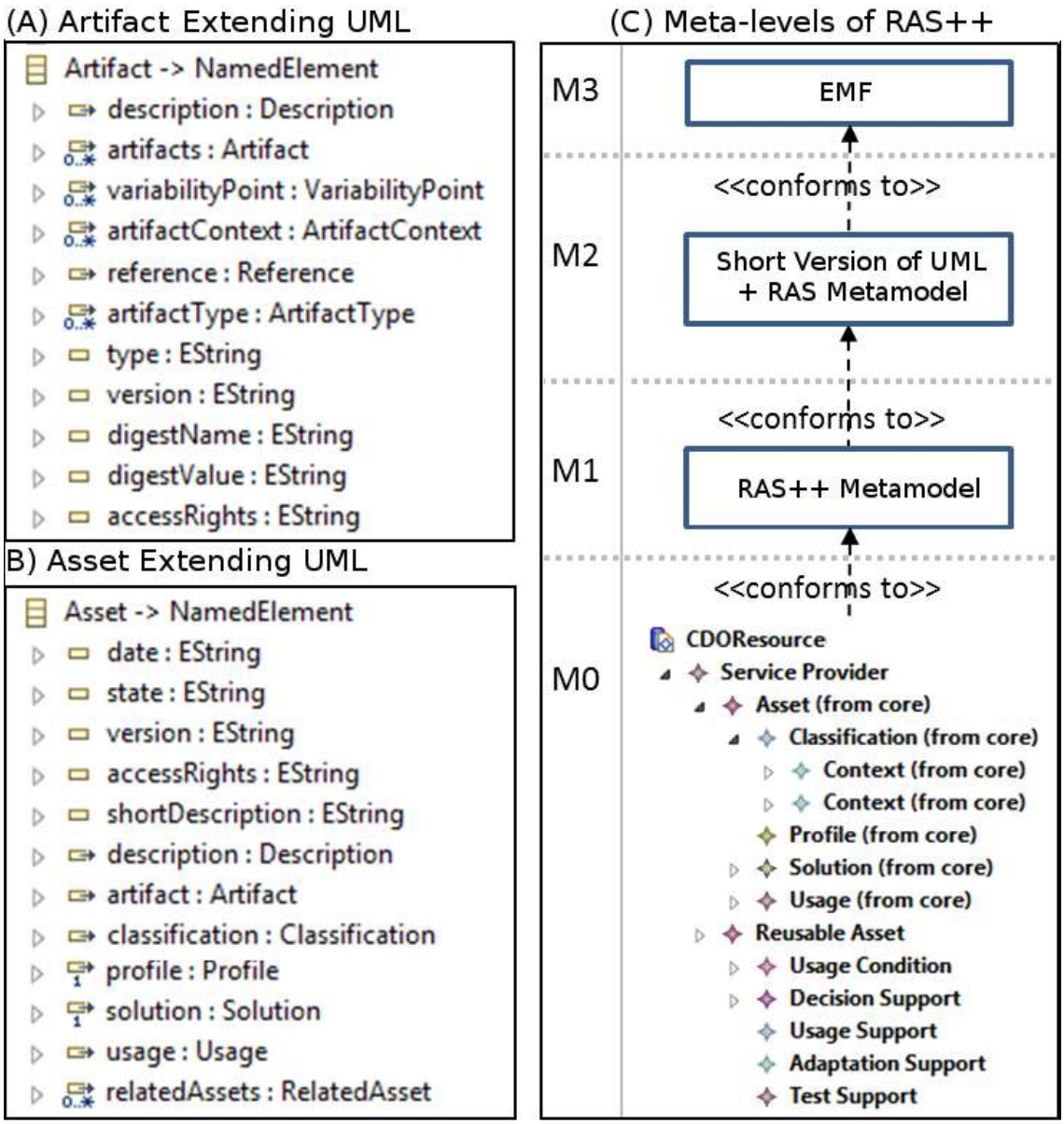


Figure 4.8: RAS++ DSL overview

Extensibility Mechanisms

Figure 4.9 shows metaclasses extracted from RAS and UML to allow extensibility in RAS++. First of all, each metaclass in RAS++ extends either `NamedElement` or `Element`. For this reason, our representation language allows two types of extensibility: 1) Through RAS Profiles; 2) Through UML Profiles.

The standard RAS definition allows the creation of new representations with the use of profiles. These extensions are heavyweight, which means that software engineers must change the definition of the XML schema of RAS. Differently, RAS++ is a DSL in conformity with EMF, which means that heavyweight extensions need the generation of a new version of RAS++ Eclipse plug-in. This approach is posi-

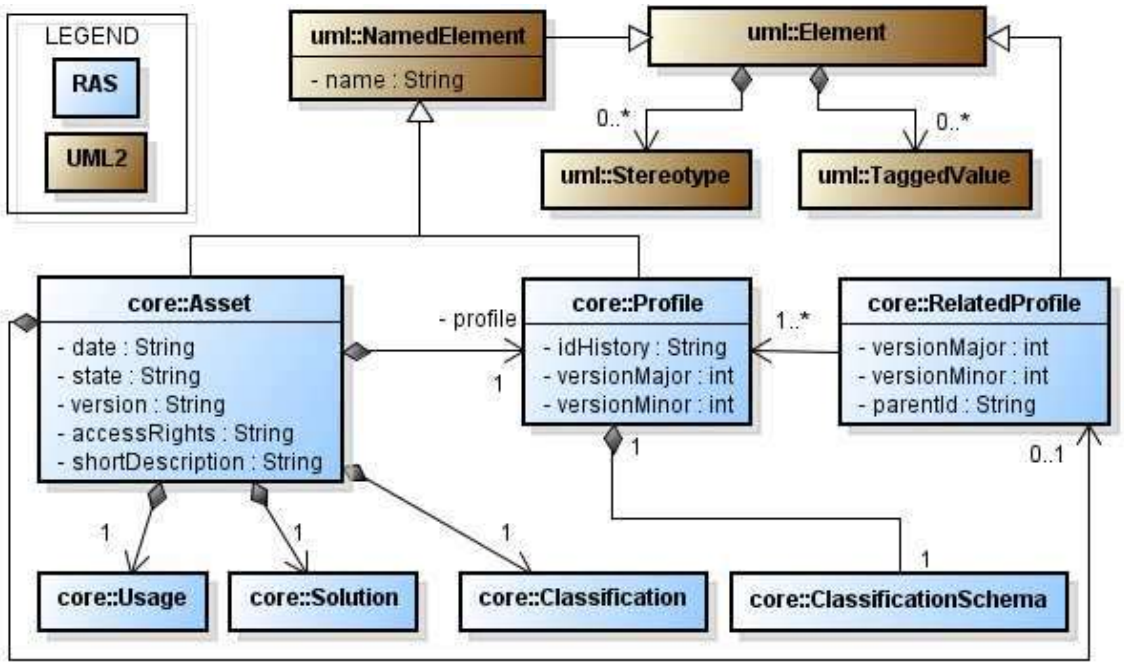


Figure 4.9: Extensibility mechanisms in RAS++

tive to keep the core language separate from profiles. As illustrated in Figure 4.9, extensions through profiles include metaclasses `Profile` and `RelatedProfile`.

A big difference between the standard RAS and RAS++ is that our language extends also a simplified UML metamodel. For example, the metaclass `Artifact` and `Asset` shown in Figure 4.8 (B) and (C) extends `NamedElement`, a metaclass from UML included as a package in RAS++. Besides, each metaclass from RAS++ extends at least the `Element` metaclass, that owns associations with `TaggedValue`, `Stereotype`, `Comment` and `Constraint`.

By enhancing UML, it is possible to specify annotations as instances of `Stereotypes` and `TaggedValue` in each model element. This allows to extend asset representations based on well accepted lightweight mechanisms. The extension mechanism through RAS Profiles is a heavyweight approach. It is more difficult since it needs the inclusion of a new package in the ECORE model of RAS++, or extension through a new plug-in, each one needing recompilation of the metamodel library. Thus, lightweight mechanisms are simpler.

Applying Property Selection Criteria

This section reports some changes and differences between RAS++ and the metamodels from RAS and AMS. Since these representations are proposed to implement repository based reuse, this section is quite technical, since it requires from the reader a certain experience in previous specification languages. The term *Specification* used to characterize this chapter suggests this technical effort: it is needed to

Icon	Metaclass	Description
	Repository	- Groups a set of assets, linking to a real repository
	Reusable Asset	- Provides descriptive and technical data for artifacts
	RelatedAsset	- Ralates two or more assets from a repository
	Classification	- Groups elements for classification of assets
	Profile	- Groups elements for heavyweight extensions
	Solution	- Groups elements that detail artifacts
	Usage	- Groups intructive elements supporting artifacts
	Context	- Provides a context of a classification
	DescriptorGroup	- Groups values and sub-groups for classification
	FreeFormValue	- A free text (non constrained for a structure)
	Role	- Provides user context on the use of some artifact
	Licence	- Informs the license of artifacts
	Artifact	- Represents general abstractions from an artifact
	Model	- A subtype of artifact to detail model abstractions
	DeployInfo	- Abstraction about the artifact deployment
	DownloadInfo	- Abstraction about the artifact protocol access
	VariabilityPoint	- Represents a point of the artifact for customization
	VariabilityPointBinding	- Links a customization with an activity
	ArtifactActivity	- Used to provide instructions for a specific artifact
	Activity	- Organizes a descriptive instruction by roles
	Description	- Free form description allowed in any element
	Path	- Provides a proxy mechanism to locate a resource

Figure 4.10: RAS++ DSL in support for Specification phase.

match specifications in a common sense.

For instance, RAS++ is built initially on a RAS metamodel. This required changes along these years to remove ambiguity introduced in artifact representations by considering a merge with AMS. Thus, as illustrates Figure 4.11, we removed from the metaclass *Artifact* the following properties: `diggestName`, `diggestValue`, and `type` (or `ArtifactType` in version 1 of RAS++).

Inspired by AMS success, we included in RAS++ some structures to allow the representation of artifacts in different locations. This allows the use of RAS++ to instantiate cloud repositories built on AMS, as discussed in an ongoing work[1].

[1]Ongoing work: Federation of COTS for MDE With a Pivotal Representation Language

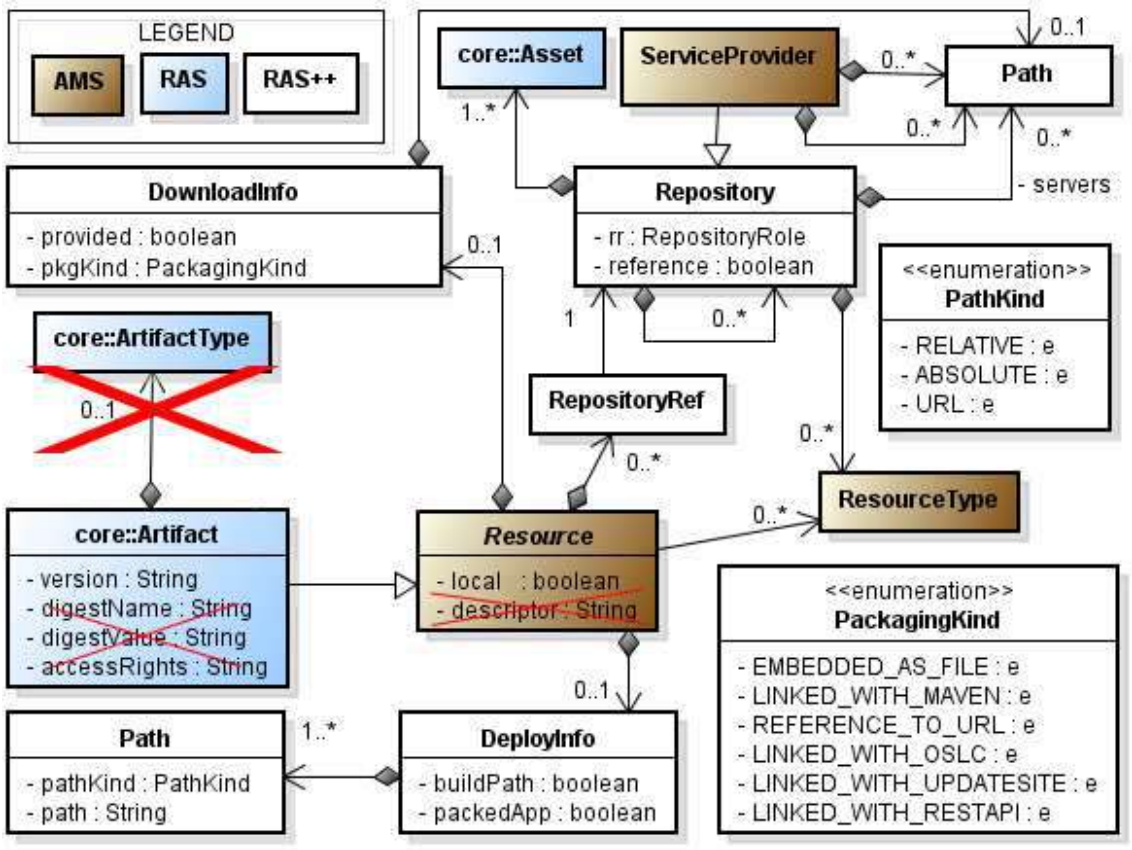


Figure 4.11: Metaclasses for artifacts and resource locators.

The removed properties are now represented with specific metaclasses: `diggestName` is replace by a property `name` available in the metaclass `NamedElement` (from the UML); `diggestValue` is replaced by a metaclass called `Resource` and; `accessRigths` is replaced by the metaclass `License`.

For artifact federation, RAS++ includes new metaclasses not exactly as those found in AMS and RAS, but that are a common sense in representations of assets: `DownloadInfo`, `DeployInfo` and `Path`. In the following, we present these changes and increments. The RAS documentation suggests to pack physical files, referenced in artifacts, as a zip file together with a MANIFEST file. MANIFEST is an XML document that represents the information exemplified in this document with asset models. Repositories unpack this zip file and store its content into databases and vice-verse. Therefore, because the standard RAS always considers a unique repository, the reference for physical files is made with absolute or with relative paths.

This is a dated mechanism for artifact federation, but RAS++ still allows its usage. In other words, the current scenario needs the federation of artifacts in distributed repositories[2]. For example, for approaches that adopt the AndroMDA toolbox as a model transformation engine (MONTEIRO *et al.*, 2014a), a set of

`prisma.cos.ufrj.br/wct/tr04.pdf`

[2]Federation- <https://wiki.openstack.org/wiki/Inter_Cloud_Resource_Federation>

libraries associated are stored in a repository that allows remote download through the MAVEN tool[3]. Other approaches store and instantiate MDE toolboxes on the cloud (BASCIANI *et al.*, 2014a), thus needing federation of resources through the REST API[4].

Other changes include: This is inspired in the AMS definition of the concepts of resource that references other resources in different service providers.

1. `Artifact` now extends `Resource`, a metaclasses extracted from AMS, with properties for locators/proxies for physical files external to the repository whose assets are stored;

2. The property `descriptor` from `Resource` (a metaclass from the AMS) is as textual property where a script for "deployment descriptor" can be inserted. Deployment descriptors should be represented with instances of `DeployInfo`;

3. The association with `ResourceType` allows classifying artifacts. We adopted this nomenclature and not `Type` as recommended in AMS, because `Type` is used in the short UML metamodel included in RAS++ to represent data types, classes and etc.

4. The unique way to represent artifact types is through instances of `ResourceType`, which must be inserted into instances of `Repository`. In standard RAS, a textual property called `type` is used, allowing mistakes and ambiguities when classifying artifacts. Thus, this definition standardizes artifact classification in RAS++, reducing mistakes;

5. The association of resources with `RepositoryRef` allows to link repositories where artifacts are stored, and;

6. Through associations with `DownloadInfo` and `DeployInfo`, a resource is composed by an information of download and deploy/installation.

The metaclass *DownloadInfo* represents where the physical file associated with the artifact is located. Three properties provide detailed information to allow the download of a file: 1) the property `provided` is used by references for MAVEN repositories and a value assigned for "true" means that the file should not be downloaded; 2) the association with the metaclass `Path` allows to represent the complete URI of the physical file; and 3) the property `pkgKind` allows to represent the semantics of access for specific types of repositories. This way, literals from the enumeration *PackagingKind* represent:

1. *LINKED_WITH_MAVEN*, to download the referenced file using the Maven tool;

2. *REFERENCE_TO_URL*, to download a file shared on the web for public access;

[3]Apache Maven - Av. at <http://maven.apache.org/> At 15/03/2016.
[4]REST API Tutorial - <http://www.restapitutorial.com/> At 15/03/2016.

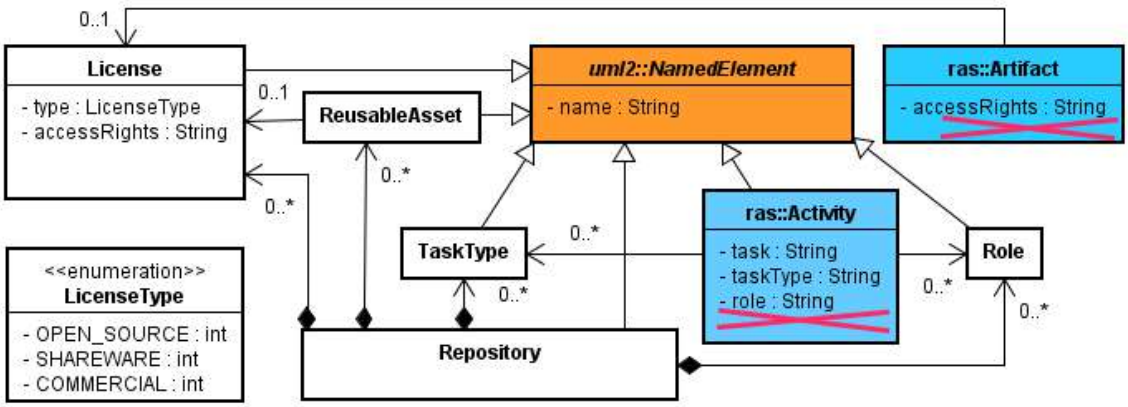


Figure 4.12: Properties from standard RAS replaced by metaclasses.

3. *LINKED_WITH_OSLC*, to download a resource from an OSLC service provider;

4. *LINKED_WITH_UPDATESITE*, the reference is an Eclipse plug-in and should be installed instead of downloaded;

5. *LINKED_WITH_RESTAPI*, the resource is downloaded using the REST API;

6. *LINKED_WITH_MODREP*, the resource is stored in a model repository, needing a specific API to connect and download the content.

The metaclass *DeployInfo* represents where the physical file must be deployed in a development environment. For example, libraries can be deployed in a specific folder called "libs", models used by model transformations are placed in a folder called "res" and model transformation test cases in a build path called "tests". The name of the folder is represented with an instance of `Path`. For folders that stores libraries that must be placed in build-paths, e.g., the "lib" folder, the property `buildPath` must be set to true. Besides, for resources such as source code that must be placed in default build-paths such as "src", the property `packedApp` must be set to true.

Figure 4.12 presents some changes in comparison to the standard RAS metamodel. The property `accessRights` is removed from the inside of `ras::Artifact` and is replaced by an association with `License`. License is also possible to be assigned by assets, in association between `ReusableAsset` and `License`. Properties from `ras::Activity` were removed and replaced by associations with `Role` and `TaskType`.

These changes are positive to reduce the number of redundant data that we observed along representations of assets. Besides, by making these properties metaclasses that extend `uml2::NamedElement`, we are allowing their extensions with lightweight UML mechanisms. Once common representations are found and the metamodel changed, the second version of RAS++ was generated without ambigu-

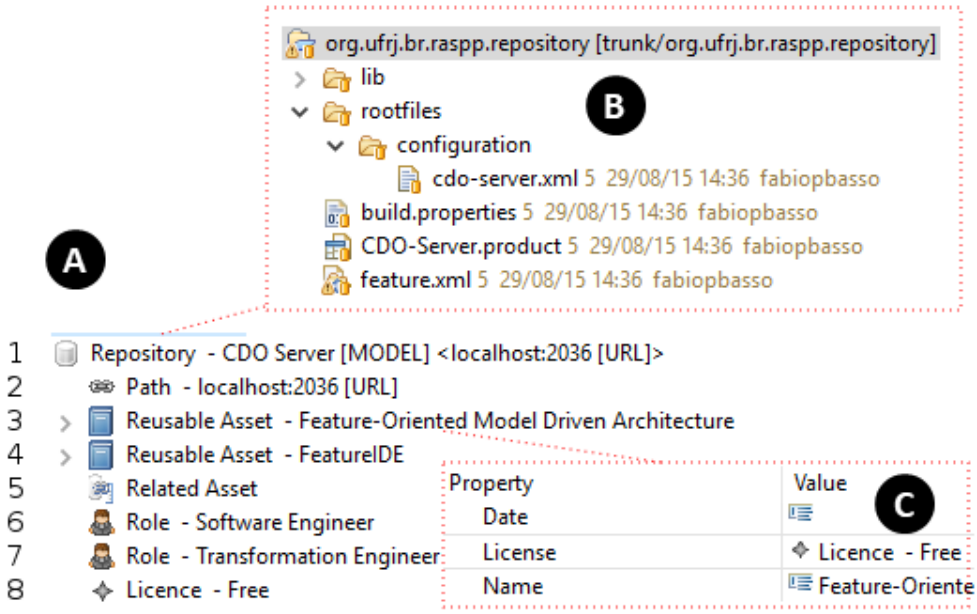


Figure 4.13: A repository describing assets and properties in RAS++

ous properties. Therefore, RAS++ fits well for representation of asset data conforms two industrial initiatives: RAS and AMS.

Demonstration in RAS++

Following some tips for representations of software components (PALUDO *et al.*, 2011), we applied RAS++ on the representation of a toolbox called Feature-Oriented Model Driven Architecture. Figure 4.13.A exemplifies a model designed conforms to RAS++. It is our practice to start with the representation of the model with an instance of `Repository`. This instance represents data for connection with some repository. This means that all the content is already stored or will be stored in the described repository. For example, the representation in lines 1 and 2 allows automatic connections with a model repository called CDO (Figure 4.13.B), which is hosted in port 2036. Thereby, these representations are not only documentation: in a proof of concept (BASSO *et al.*, 2014b), we could automatically publish asset representations shown in lines 3 and 4 in a relational database. This tool support is complementary to this thesis, thus non appropriately evaluated and detailed. As a long term goal, with new advances in tool support, we hope to extend this possibility in integrations with MDE repositories such as ReMoDD, SEMAT and GEMOC.

In order to demonstrate the application of concepts introduced in Figure 4.12, Figure 4.13.A shows two instances of `Role` and one of `License` (lines 6-8). Roles and licenses are managed by an instance of `Repository`, reducing the possibility for introducing redundant data. In RAS, this data is replied in each artifact and asset, while in RAS++ this data is provided through associations, as illustrated in

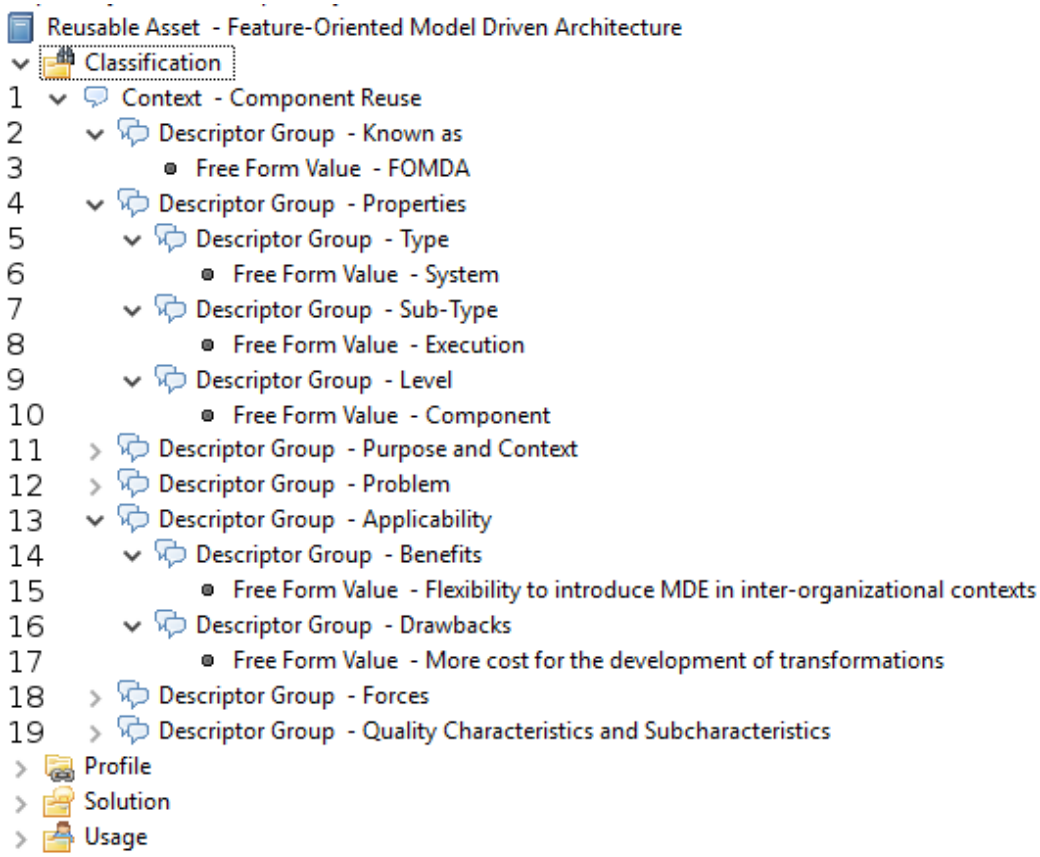


Figure 4.14: Asset describing the FOMDA Toolbox (Part I)

Figure 4.13.C.

Figure 4.14 shows RAS++ properties applied to represent an asset from our research group: the FOMDA Toolbox. Since there is no guideline for representation of assets describing toolboxes, we used 19 structures suggested in (PALUDO *et al.*, 2011). These structures are for the description of software components, thus characterizing our asset as "Component Reuse" (instance of `Context` in line 1). Lines (2, 4, 5, 7, 9, 11-14, 16, 18 and 19) demonstrates that, through instances of `DecriptorGroup`, it is possible to represent 12 structures from the 19 suggested, detailing each one with textual information allowed in instances of `FreeFormValue` (lines 3, 6, 8, 10, 15 and 17).

The other seven structures suggested in (PALUDO *et al.*, 2011) are represented with other metaclasses: Figure 4.15 presents descriptions for solution, and Figure 4.16 for usage. These metaclasses follow the RAS standard, discussed in details previously, and match completely the needs discussed by Paludo. For example, Figure 4.15 shows details represented in instances of `Artifact`[5] in lines 1, 6, 14, 16 and 17. The first artifact is a "Java Library". It is physically stored in a MAVEN repository, shown in line 3, and must be deployed in a relative path "

[5]Model is an instance of Artifact discussed in the Transformation phase

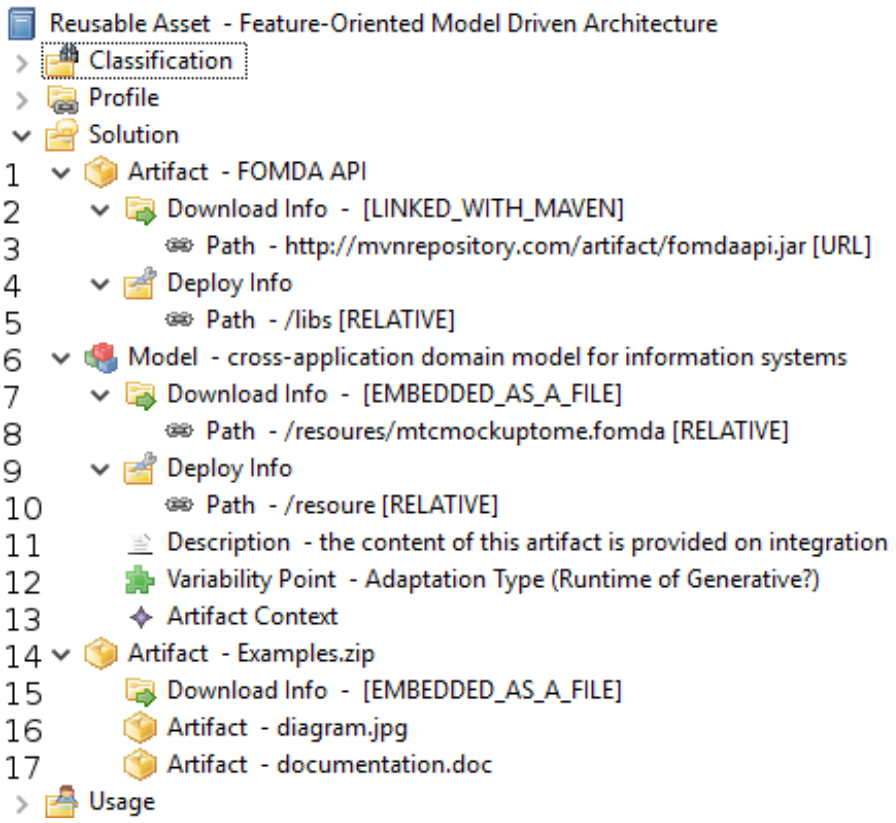


Figure 4.15: Asset describing the FOMDA Toolbox (Part II)

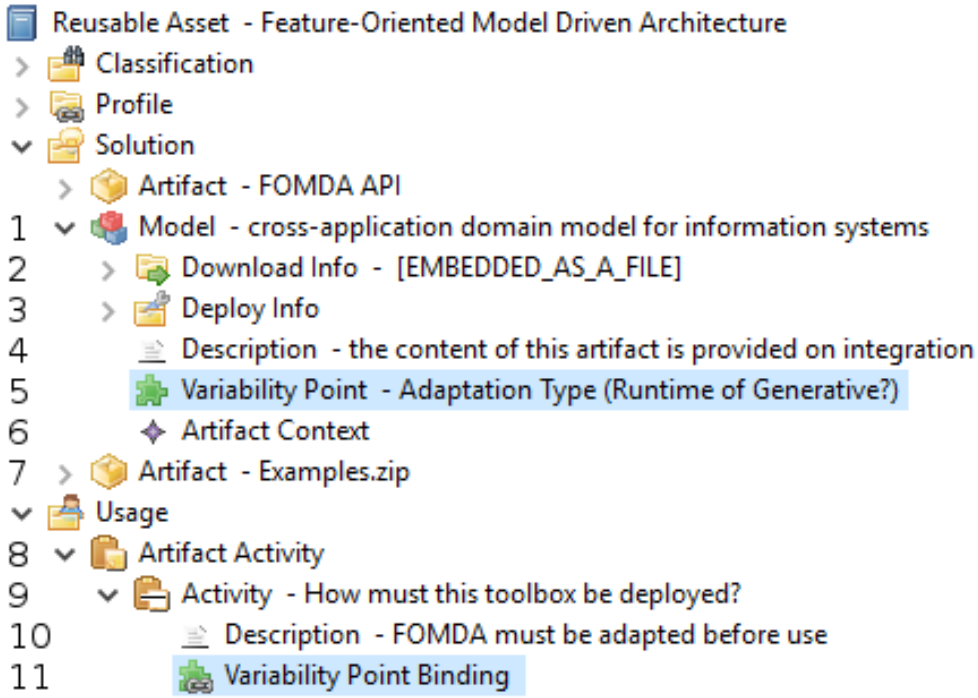


Figure 4.16: Asset describing the FOMDA Toolbox (Part III)

libs". This is possible by using resource locators composed by instances of `DownloadInfo` (lines 2, 7 and 15) and `DeployInfo` (lines 4 and 9).

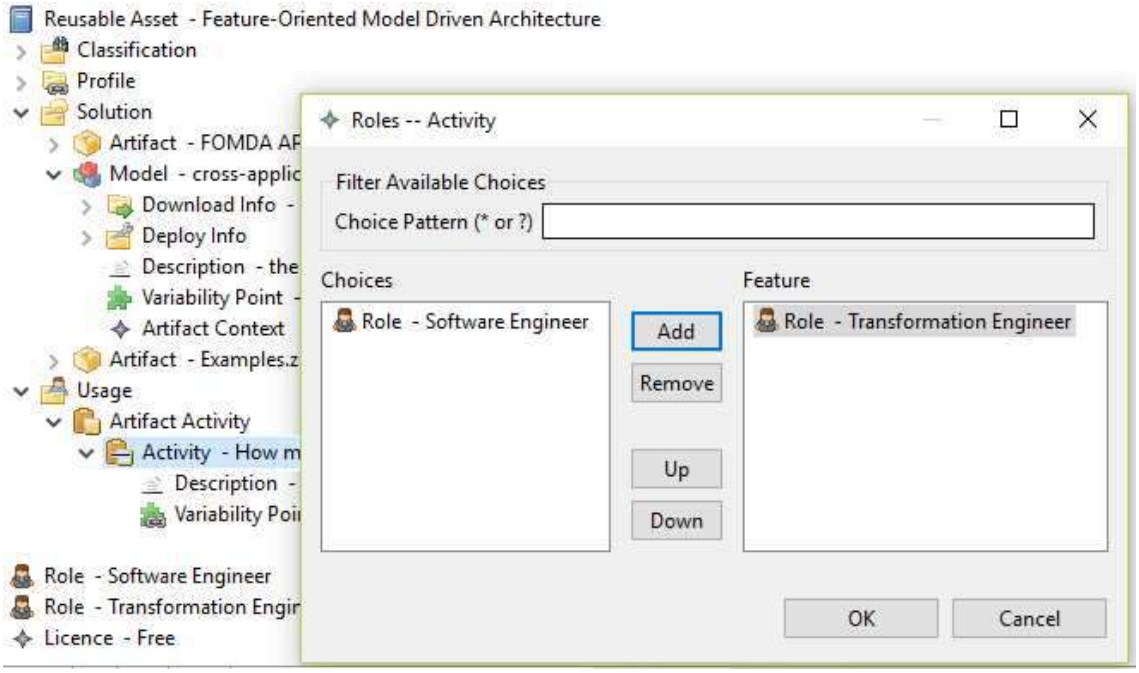


Figure 4.17: Association of roles into activities for artifacts

Figure 4.17 shows a screen-shot of a dialog available in our tool support for representation of RAS++ assets. In this case, we are associating the role "Transformation Engineer" into the activity "How must this toolbox be deployed?". This association establishes the following semantics: this role is responsible for execution of the activity "How must this toolbox be deployed?". This activity is also associated with a variability point shown in Figure 4.16 (line 5) through an instance of `VariabilityPointBinding` (line 11). Semantically, this means that the "Transformation Engineer" should customize the artifact "Model - cross-application domain model for information systems" shown in line 1. This rich semantic allowed by RAS is one of the reasons why we selected it as basis for contributions of RAS++.

4.1.4 Final Remarks

Table 4.1 compares general properties from RAS, AMS, and RAS++. In order to match the needs in MDE as a Service, as well as modernize RAS to novelties available in AMS (e.g., artifact federation), RAS++ merges these specifications. As can be observed, RAS++ is more representative than the other two, which is a point in favor to enable interchange of data as a pivot. We demonstrated in this chapter the representation of an asset describing an isolated toolbox. This is limited considering our motivation: tool chain, a contribution that we depicted later. Moreover, the implementation of coopetiton in MDE as a Service requires more than matching these asset specifications. Therefore, the next chapter presents our consideration regarding the quality of the descriptive information by depicting

Table 4.1: Comparing 26 properties from RAS, AMS and RAS++

Property	RAS	AMS	RAS++	Property	RAS	AMS	RAS++
1. Representation format	XML	RDF	XMI	14. License	Yes	No	Yes
2. Asset Relationships	Yes	Yes	Yes	15. Version	Yes	No	Yes
3. Represent Any Artifact	Yes	Yes	Yes	16. Services	Yes	Yes	Yes
4. Artifact Federation	L	D	D	17. Repository/Service Provider	No	Yes	Yes
5. Light-weight Extensibility	No	No	Yes	18. Variability Point	Yes	No	Yes
6. Asset Profile	Yes	No	Yes	19. Variability Point Binding	Yes	No	Yes
7. Descriptor group	Yes	No	Yes	20. Activity Parameter	Yes	No	Yes
8. Classification	Yes	Yes	Yes	21. Artifact Activity	Yes	No	Yes
9. Free form value	Yes	Yes	Yes	22. Activity	Yes	No	Yes
10. Context	Yes	No	Yes	23. Resource Type	No	Yes	Yes
11. Classification Schema	Yes	No	Yes	24. Deploy Info	No	No	Yes
12. Node Descriptor	Yes	No	Yes	25. Download Info	No	No	Yes
13. Free Form Descriptor	Yes	No	Yes	26. Path/URI/URL	Yes	Yes	Yes

an important preliminary phase for tool chain: the *Acquisition*.

4.2 Asset Acquisition

Previous sections presented common properties in support for the *Specification* phase, shown in Figure 4.18 (A). In this sense, it was exemplified an asset describing a toolbox using structures recommended from software component context (PALUDO *et al.*, 2011), as illustrates Figure 4.18.B. In this sense, an open question is: would these structures, suggested on the reuse of software components (PALUDO *et al.*, 2011), be adequate to take good decision about the acquisition of some assets shared in the context of MDE as a Service? Some works suggested limitations for decision making in the state of the art (AXELSSON *et al.*, 2014; BADAMPUDI *et al.*, 2016) since there is no criteria for comparison of MDE Artifacts in the *Acquisition* phase. Therefore, the answer to the aforementioned question is no.

Recent studies agree that finding the right options for specific needs is a requisite for better introducing MDE Artifacts in target software projects (COMBEMALE *et al.*, 2014). In this direction, similarly as in tool chain integration (BATORY *et al.*, 2013b), the success of MDE as a Service in a coopetition scenario relies on finding the right options for specific requests from target contexts (MUSSBACHER *et al.*, 2014). This is an issue for decision making, lacking a solution that helps software engineers in a preliminary phase for tool chain (BIEHL *et al.*, 2014), as motivated recently (ZAKHEIM, 2017).

In a previous work, we found that this need resembles to the descriptive information represented in a structure that allows analytical and computational comparisons (BASSO *et al.*, 2016b). Repositories proposed several ways for providing descriptive data. However, we missed an appropriate representation for the criteria found for the specificity of MDE Artifacts and Settings (HEBIG *et al.*, 2013), thus opening a call for action (BASSO *et al.*, 2015a) towards integration of some prin-

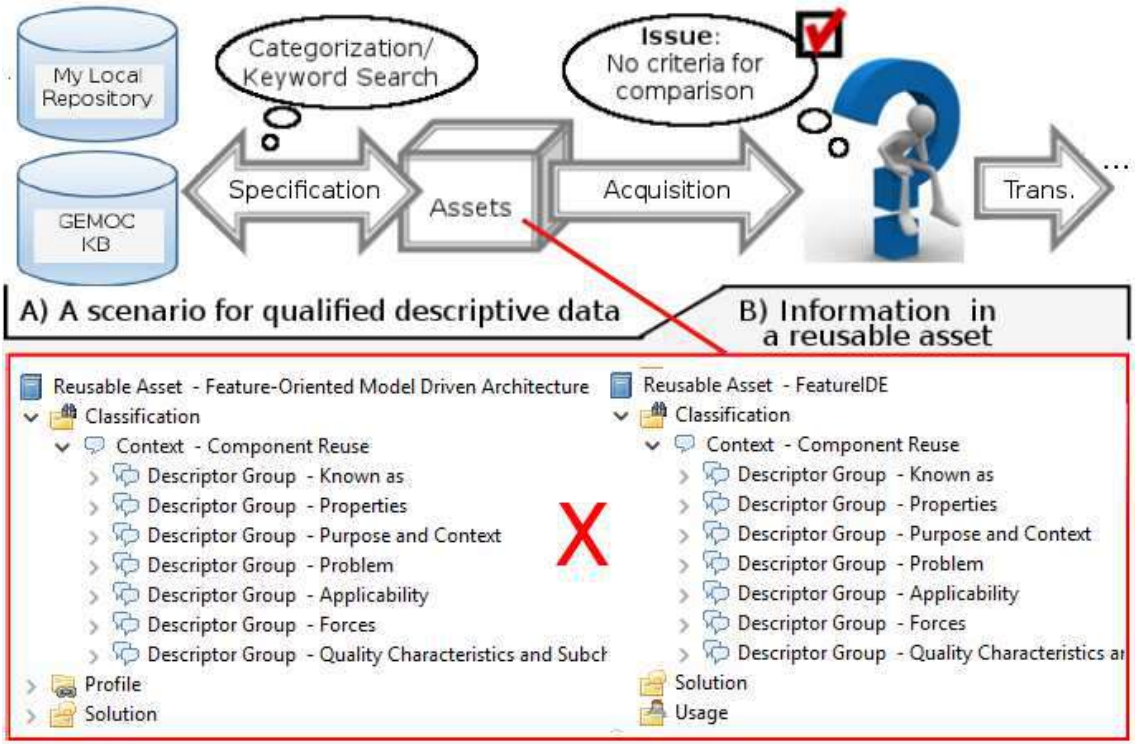


Figure 4.18: Decision making in the Acquisition phase.

ciples from Software Ecosystem (SECO) (SANTOS and WERNER, 2012) with the MDE specificity (MDE Ecosystems).

In the research topic "Acquisition", SECO platforms have been proposing mechanisms for negotiation (MANIKAS, 2016; MENGERINK *et al.*, 2016; OSLC, 2017b; SANTOS *et al.*, 2016). These platforms are very important for enabling coopetition, where asset consumers interact with asset producers in a business environment. Although business and negotiation are critical for implementation of coopetition, we are investigating a more fundamental issue: the lack of qualified data in assets hampers the execution of the *Acquisition* phase. Thus, we believe that the best SECO platform will not be so effective if assets are not represented with quality.

This section is organized as follows: Section 4.2.1 presents the research method employed to derive criteria for representation of asset, established in support for coopetition. Section 4.2.2 presents an analytical study on the application of the established criteria to understand benefits and limitations from asset specifications. Section 4.2.3 introduces the RAS++ metamodel and an example of representation for validation of descriptive data from assets. Section 4.2.4 summarizes our conclusions.

4.2.1 Deriving Criteria for Qualified Data

We investigated whether assets represent a good alternative to be used in this preliminary phase of tool chain for MDE Artifacts and Settings. Likewise, we found that the problem for using assets in MDE repositories is twofold: 1) we miss criteria to represent this qualified data with enough semantic potential to support reuse of MDE Artifacts globally, and; 2) we miss the support for reuse trough an appropriate representation language mapped for such criteria. In other words, the literature of the area is limited in terms of representations used independently from a specific repository vendor and the proposed representation languages for assets do not capture the data reported as relevant for decision making along the selection of MDE Artifacts from repositories.

Goal

Our main goal is to answer the research question: **Q5.2): What is suggested in the literature as quality attributes for representation of assets?** Our goal is to provide a list of properties for representation of qualified data in assets. Since many structures are available in repositories such as ReMoDD and SEMAT, this list should be used as a quality criteria rather than as a structure for representation.

Our goal in this section, therefore, is to derive a criteria for representation of assets. Our motivation is that MDE assets as the one shown in Figure 4.18 (B) are currently specified following an ad-hoc strategy, thus hampering the reuse in global scale and making less feasible the implementation of coopetition in a near future.

Research Method

In order to answer the first research question, we executed four ad-hoc literature reviews and five mapping studies (MARTINEZ-FERNANDEZ *et al.*, 2015), as described in Table 4.2. After extracting criteria for representation, we executed an analytical study (YIN, 2003), thus answering the second research question.

Execution

This section is contextualized in the research process shown in Figure 4.19 that organizes our contributions associated with RAS++ in five states. The contributions are listed in Table 4.2. Our criteria is derived in state 3, which means that we conducted many preliminary studies: from a research idea in state 0, we derived some technical contributions for MDE Artifacts and Assets in state 1, resulting in a central issue (state 1.4); then, we divided two lines of research, from 2.1 to 2.3 to find descriptive data and from 2.4 to 2.6 to find technical data; state 4 is

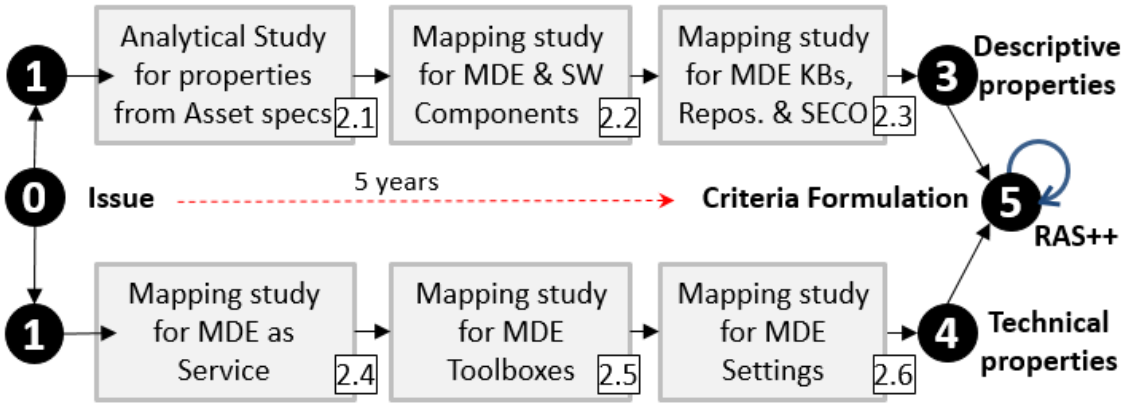


Figure 4.19: The process adopted to derive criteria

therefore characterized by a clustering study executed with a coding protocol from mapping studies 2.4 to 2.6. State 5 is reached through the clustering studies in 4 and 3, resulting in contributions for RAS++. State 3 is therefore characterized by a second clustering study from 2.1 to 2.3, resulting in a set of elements for the description of MDE Artifacts.

After state 2, we reasoned that these analytical and mapping studies are limited to the material published in search engines. So, we acknowledge the existence of other data sources not found in keyword search. For example, to reach better trust in state 3, we missed from studies 2.1 to 2.6 properties that could be used for decision making in an asset selection process, as previously motivated. Then we conducted a multivocal literature review (OGAWA and MALEN, 1991), which is mainly characterized by the inclusion of material not available in search engines. This section therefore extends a previous work (BASSO *et al.*, 2016b) by including new sources of data and revisiting data extracted previously in state 2.3. Hereafter, we found that the subject of research is new, needing another approach for data elicitation.

The study 2.3, executed through a multivocal research mapping[6], adopts the following procedure:

- searched in books about reuse in global scale for MDE (COMBEMALE *et al.*, 2015b), design patterns (GAMMA *et al.*, 1995) and empirical software engineering (HUTCHINSON *et al.*, 2011);
- searched in contributions scoping mapping studies for software ecosystems (AXELSSON *et al.*, 2014);
- searched in papers on management of model-based operations (KUSEL *et al.*, 2015);

[6]Multivocal - A multivocal research considers other source of data besides key-wording in searching engines like Scopus.

Table 4.2: Organization of our studies for criteria formulation and for implementation of RAS++

State	Summarized references
0	How do you Execute Reuse Tasks Among Tools? A RAS Based Approach to Interoperate Reuse Assistants
1.1	A Common Representation for Reuse Assistants
1.2	Supporting Large Scale Model Transformation Reuse
1.3	Towards Facilities to Introduce Solutions for MDE in Development Environments with Reusable Assets
1.4	A Proposal for a Common Representation Language for MDE Artifacts and Settings
2.1	Analysis of Asset Specification Languages for Representation of Descriptive Data from MDE Artifacts
2.2	Semantic Properties of Software Components: A Systematic Mapping Study
2.3	Intents from Asset Platforms and Their Properties: A Multivocal Mapping Study
2.4	Characterizing the "MDE as Service" Research Agenda
2.5	Diversity of MDE Toolboxes and Their Uncommon Properties
2.6	MDE Setting Intents and Their Properties: A Systematic Mapping Study
3	Criteria for Description of MDE Artifacts
4	Criteria for Representation of Technicalities from MDE Artifacts
5.1	Extending RAS to Better Describe MDE Artifacts (see Chapter 4.1)
5.2	Extending RAS to Represent MDE Artifacts and Their Settings (see Chapter 4.3)
5.3	This Doctoral Thesis

- in technical reports from industry such as from Tasktop tool chain teams (KERSTEN, 2013; ZAKHEIM, 2017), Gartner Group evaluating assets (GARTNER, 2013), OSLC partners discussing about assets for application lifecycle management (OSLC, 2017b); and

- looked for calls for contributions (tool demo) from conferences related to MDE (MODELS, ECMFA, SPLC, GPCE, SLE, etc.) and in more general conferences (ASE, ICSE, OOPSLA) to understand the criteria considered by reviewers to decide whether a tool is relevant.

Through the mappings 2.1 and 2.2, it was found quality attributes for assets focused in Component-Based Software Development (CBSD) (HONG-MIN *et al.*, 2010), but with no significant differences from attributes presented by (PALUDO *et al.*, 2011). In addition, studies 2.3 to 2.6 resulted in data recommended for selection of toolboxes considering business, social and technical perspectives. This analytical process took five years, resulting in properties relevant for decision making in MDE as a Service scenarios.

Analysis

This emerging field scoped in MDE as a Service is both broad and diverse, so it is difficult for practitioners to understand how to analyze adoption opportunities and for researchers to reach inter-relations between toolboxes used in different phases for tool chain. Likewise, quality attributes derived from well formed/structured and detailed descriptions are the way to facilitate decision making about the asset that best meets a customer's expectations (BADAMPUDI *et al.*, 2016). For

example, Figure 4.20 illustrates four assets with descriptive data. So far, it is not acknowledged which criteria should be considered for these representations, implying in ad-hoc strategies for description. In this sense, this section consider some quality attributes discussed by the literature of the area by means of representation criteria, thus answering **Q5.2: What is suggested in the literature as quality attributes for representation of assets?**

Figure 4.20 shows side-by-side four assets in the context of SPL. For didactic purposes, these assets describe toolboxes found in our research group as follows: A) an asset describing FOMDA Toolbox (BASSO *et al.*, 2013a); B) an asset describing UML-FI Toolbox (OLIVEIRA *et al.*, 2011); C) an asset describing Odyssey-Arch Toolbox (FERNANDES *et al.*, 2011), and; D) an asset describing artifacts for model transformations in support for the MockupToME Toolbox (BASSO *et al.*, 2016d). Toolboxes are contextualized as supporting MDE features.

General quality attributes for description of assets are clearness, synthesizing power, unambiguity, correctness, structural power, completeness and traceability of dependencies, which can be internal to a repository or external. In the following, we present criteria important for asset producers to ensure that these quality attributes are satisfied.

C1 - Presentation Features

A presentation latter should synthesize presentation data since this is the first information read by asset consumers. The following questions help asset producer to apply such attributes in asset descriptions: 1.1) What is the solution? 1.2) How to use it? 1.3) Who can use it? 1.4) Known uses? It is also important to save asset consumers time when analyzing assets, including the following considerations: 1.5) What the solution is not? 1.6) When avoid/reject it? 1.7) Who should avoid it? 1.8) Which are the known incompatibilities?

C2 - Usage Features

Questions recommended by empirical engineering are important to establish comparisons between competitor MDE Artifacts, such as:

- **2.1) Requirements -** Which are the minimum requirements to use an artifact? One should include data about team configuration, members' skills (social and technical) which the artifact have been used successfully, the required knowledge from end-users about domain specific languages, with which software process, which phases of software development, and so on.

- **2.2) Round-up data -** Which are the pre, and post-conditions to use it? Which are the benefits and drawbacks? dos and don'ts, pros and cons.

- **2.3) Success and unsuccessful stories**, reporting which are the evidences that support the solution and in which boundary conditions?

Figure 4.20: Assets with structural features for description allowed in RAS.

- **2.4) General considerations -** software component reuse use to describe reuse opportunities through the following questions: Which is the license rights? Which are the asset dependencies? Which are the reuse opportunities already established? Which reuse opportunities are expected in the future?

C3 - Well-formedness Features

It is needed to take care with textual data of type "Free Form" (FRANCE *et al.*, 2006). In this sense, this criteria focuses in the correctness and unambiguity attributes. Well formed texts are possible of verification from two different perspectives: from the asset provider and from the repository maintainer.

- The following quality attributes help asset providers to keep the "well-formedness" in asset classification: **3.1) Unambiguity -** the information represented textually should not be repeated along classifications and sub-classifications. **3.2) Correctness -** for immutable values, it is expected that a "free form value" is identical to a set of possible values provided by the repository.

- It is also responsibility from repository maintainers to keep well-formedness rules in support for descriptive data. This way, the following questions should be considered by repository maintainers: **3.3) Standard taxonomy -** Is the information for cataloging based in standard taxonomy from the artifact domain? Is this taxonomy representative? Is this taxonomy designed in an empirical investigation? **3.4) Data dictionary -** Is this standard taxonomy represented as data dictionary in the repository?

C4 - Business Features

It is important to provide data about business opportunities (AXELSSON *et al.*, 2014) and asset perspectives (GARTNER, 2013). In this sense, following properties should be represented in a business perspective:

- **4.1) Deployment efforts**, including the learning curve and the Return on Investment (ROI) (WHITTLE *et al.*, 2015).

- **4.2) Cost properties** could consider the structure (development cost, product cost, operating cost and information cost) AXELSSON *et al.* (2014).

- **4.3) Feedback from users** (HADJI *et al.*, 2008) classified according to customer segments (broader customer reach, improved customer feedback and customer differentiation) (AXELSSON *et al.*, 2014), including free form descriptions represented in user groups and social networks (LIMA *et al.*, 2016).

- **4.4) Actors** associated with the artifact (manufacturer, supplier, content provider, communication provider, service operator, add-on developer, owner, end user, regulatory agency, information broker) (AXELSSON *et al.*, 2014).

- **4.5) Business opportunity links** (customer segments, value proposition, channels, customer relationships, revenue streams, key resources, key activities and key partners) (AXELSSON *et al.*, 2014).

- **4.6) Asset lifetime stage -** The relevance for the asset lifecycle phase (requisition, procurement, deployment, maintenance and retirement) (GARTNER, 2013).

- **4.7) The goals** for artifacts such as future evolutions, integrations, probable deprecations.

C5 - Clustering Features

The creation of descriptor groups (OMG, 2005) should obey some classification from the literature. In this sense, it is important to consider structural contexts adopted for description purpose, including well formed constructors for groups, values, categories and sub-categories. We found the following approaches in structural features for assets:

- **5.1) Application contexts -** This structural feature is widely used in representations of assets, but it also is dependent from an application domain. Typically, application contexts suggests the inclusion of phases of software development, which is broad and diverse in Software Engineering discipline. Thus, we found taxonomy for application contexts that can structure elements for:

  - **5.1.1)** General software artifact development contexts (FRANCE *et al.*, 2007);
  - **5.1.2)** Application Lifecycle Management (ALM) (JOHNSON *et al.*, 2012; ZAKHEIM, 2017);
  - **5.1.3)** User contexts for software components (HADJI *et al.*, 2008; PALUDO *et al.*, 2011);
  - **5.1.4)** Model transformation contexts (LÚCIO *et al.*, 2014; MENS and GORP, 2006);
  - **5.1.5)** MDE-based process contexts (CZARNECKI and HELSEN, 2003; KLEPPE *et al.*, 2003);
  - **5.1.6)** Programming language contexts (SHAN and HUA, 2006);

  - **5.1.7)** Analysis strategies that classify SPL Toolboxes (THÜM *et al.*, 2014);
  - **5.1.8)** Industrial adoption issues on MDE tool contexts (WHITTLE *et al.*, 2015);
  - **5.1.9)** MDE knowledge representation contexts (BATORY *et al.*, 2013a; DA SILVA, 2015)
  - **5.1.10)** Tool demo/research environment contexts (GORP and MAZANEK, 2011), and;
  - **5.1.11)** Tool chain tasks (BATORY *et al.*, 2013b; LIEBEL *et al.*, 2014).

- **5.2) Warnings and highlights -** Structured data that draws the user's attention to critical points for the use of shared artifacts, such as abstractions that make clear when an MDE Artifact needs adaptation before introduction in a target context (BATORY *et al.*, 2013b).

- **5.3) Asset and artifact relationships -** competing or cooperative assets should be related (AXELSSON *et al.*, 2014; BADAMPUDI *et al.*, 2016), providing structures for textual information for decision making (business, technical and social perspectives) (SANTOS *et al.*, 2016).

- **5.4) Domain specific structure -** Association of a descriptive structure with a domain/context, such as repository the case for MDE Artifacts (FRANCE *et al.*, 2007) that considers a structure better mapped than other built to support CBSD (PALUDO *et al.*, 2011).

- **5.5) Structured assessments -** Structuring textual empirical data that provide evaluation of application/usage of shared artifacts (RUNESON and HÖST, 2008), such as type (Case Study, Survey, Experiment and Action Research), research methodologies (Exploratory, Descriptive, Explanatory and Improving), among other definitions (NETO *et al.*, 2008).

- **5.6) Structured quality model -** Descriptions built on model proposed for internal and external quality (ISO/IEC, 2011) such as Functionality, Usability, Efficiency, and Maintainability, and others for the specificity of MDE toolboxes (MOHAGHEGHI and AAGEDAL, 2007).

- **5.7) Pattern catalog -** Many researches (ELGEDAWY, 2009; HAMMOUDA, 2005; HEBIG *et al.*, 2012; LANO and RAHIMI, 2014; LANO *et al.*, 2017; PALUDO *et al.*, 2011; TRAN *et al.*, 2011) consider important structuring asset representations based on patterns (GAMMA *et al.*, 1995).

C6 - Integration Features in MDE

This criteria is derived exclusively in support for MDE Artifacts. Thus, it is essential to represent elements supporting quality attributes completeness and traceability of dependencies, such as:

- **6.1) Integration instruction -** Typical MDE Artifacts will need instructions for introduction in target contexts (LIEBEL *et al.*, 2014). Is clear the instructive information, such as how to integrate and use artifacts in contexts?
- **6.2) Target platform -** The target platform associated with transformations for model-to-text/model-to-code/code-to-model (KLEPPE *et al.*, 2003).
- **6.3) Artifact intents -** asset providers should provide intents of artifacts (model, metamodels, transformations and tools) on in integration context. For example, describe the type of the model transformation by its intentions and properties (LÚCIO *et al.*, 2014).
- **6.4) Toolbox and dependencies -** For MDE Toolboxes this includes metamodels and the meta-model framework (COMBEMALE *et al.*, 2014), as well as the design languages associated with each model transformation, such as UML Profiles, or Graphical DSL, or Textual DSL (KELLY and TOLVANEN, 2008). For general toolboxes (ZAKHEIM, 2017) this includes representing artifacts for web services, component interfaces, WSDL, serialization formats.
- **6.5) Building blocks and chains -** Model transformation components, connectors, bindings and parameter matching (YIE *et al.*, 2012), integration interfaces/ports (BIEHL *et al.*, 2014), proxies for resources (ZAKHEIM, 2017).
- **6.6) Variability points -** describing adaptation needs in components, domain specific languages and tools (OMG, 2005), and;
- **6.7) Supporting material -** installation procedures, examples, how to, guidelines, integration procedures, libraries, scripts, rules, and others.

4.2.2 Analyzing Asset Specifications in Coopetition Scenarios

As demonstrated previously through an analysis of the state of the art about structural features found in asset proposals, implementation of coopetition in MDE as a Service is complex and needs data for decision making. In this sense, this section considers a new study: an analysis of metaclasses from RAS and AMS in support for the aforementioned criteria for qualified data.

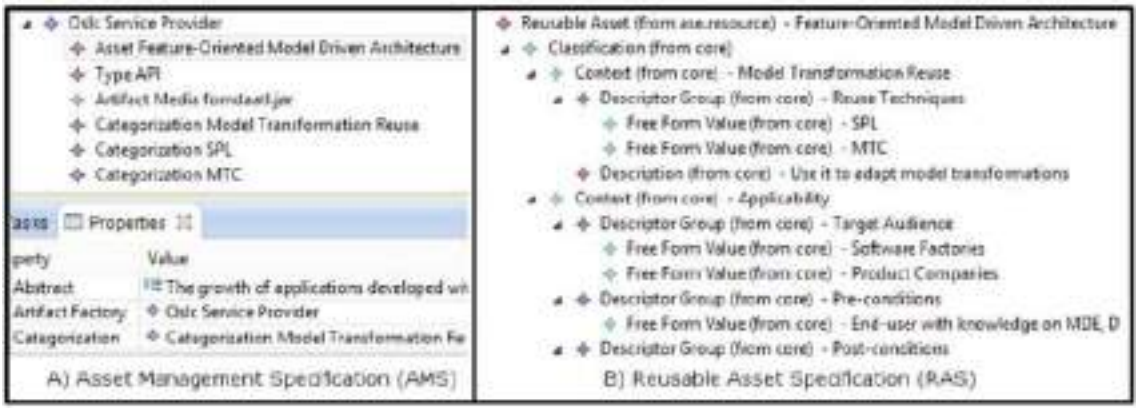


Figure 4.21: Revisiting assets from a toolbox called FOMDA.

Goal

Our main goal is to answer the research question:

- **Q5.3) Whether RAS and AMS fulfill the needs for structural features from the criteria?**
- **Q5.4) Which research gaps need to be investigated for the conception of this language for this emergent reuse scenario?**

We investigate whether RAS and AMS can represent the required information in coopetition scenarios. Based on previous criteria, we analyzed benefits and limitations in RAS and AMS, introducing some requirements for a "pivotal language" and associating specific research gaps in the area.

Research Method

We present an analytical study on the feasibility of the current technological support for a pivotal representation language for MDE through demonstration. We revisited a previous work in comparing RAS and AMS (BASSO *et al.*, 2016c), which represented 37 assets, applying the identified criteria. This criteria considers each representation language, thus using two DSLs generated with the Eclipse EMF illustrated in Figure 4.21.

Analysis

This section answers **Q5.3) Whether RAS and AMS fulfill the needs for structural features from the criteria?**

In summary, we found that both specifications provide structural features for cataloging information. Thy also 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 regarding flexibility for representation of hybrid structural features, as observed in this mapping, through metaclasses such as `Context`, `ClassificationSchema` and `FreeFormDescriptor`.

Benefits

In the following, we discuss the results of our analysis with regards to their associated 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. This means that assets can be downloaded just for the sake of understanding what it is, before deciding to use the associated content;

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-** Another advantage in specifying information in assets is that it is independent from a repository vendor. In the case of artifacts described by AMS, this information is specified with 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 can be stored in several repositories, enabling a distributed knowledge base for MDE resources. This is possible through dependence links proposed by AMS, which contain information about the target repository; and

6. **Applicable to describe many MDE resources-** Asset specifications can be used to describe any other component used in MDE context, such as any technical solution or even techniques and didactic materials for Software Engineering in general. Some initiatives are using assets specified with AMS to

describe tools provided on the cloud such as database management systems, application servers and custom applications.

Limitations

The following limitations are found in RAS and AMS in support for criteria:

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. This would benefit proposals for adaptation in artifacts considering variability in target software projects, for example; and

2. **Lack of meta-classes to represent artifacts and tasks in more technical-level-** It is important to include in assets also the information associated with model transformations and tool chains. Asset specifications have no structural support for this type of information associated with model-based components/tasks. Thus, these languages should extend the common use for search and retrieval information in repositories to enable the integration of MDE resources in target specifications for execution.

Research Gaps Considering Structural Features

We found some research gaps for conception of a Pivotal Representation Language considering descriptive data. Thus, this Section is focused in descriptive data and answers part of **Q5.4) Which research gaps need to be investigated for the conception of this language for this emergent reuse scenario?**

Specifications provide rich structures to describe MDE artifacts that are currently represented in some repositories. Both specifications present some differences and overlapping in classification information, used by repositories to catalog and retrieve artifacts associated with the described DSLs for MDE. We reasoned that RAS owns more meta-information than AMS, such as descriptor groups, free form values (i.e., long texts) and instructive information (i.e., activities). However, structures for descriptive information can be improved.

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. 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 model transformations.

Structural features are well tailored for description but not well tailored for technicalities from MDE Artifacts. RAS and AMS are general specifications, allowing classification with any data, thus well tailored for structural features found in criteria, except for C6. We miss adequate structures for representation of MDE artifacts.

Research Gaps Considering Coopetition

In the following, we present research gaps for conception of a Pivotal Representation Language considering coopetition in MDE as a Service.

This section complements the above answers with research gaps from a technical perspective regards to the representation of MDE Artifacts and Settings. Likewise, several proposals for connection MDE artifacts have emerged as DSLs for MDE Settings (HEBIG, 2014). Examples are contributions for model transformation chain (ETIEN *et al.*, 2013), component model for model transformation (CUADRADO *et al.*, 2015) and other concepts for processes (MACIEL *et al.*, 2013) and reuse (BASSO *et al.*, 2013a). However, few is acknowledged about requirements for representations of disconnected artifacts, shared and reused independently from processes and components. Thus, one has to consider that in a coopetition scenario illustrated in Figure 4.22, MDE artifacts are not acknowledged "a priori" by reusers, thus needing extra information discussed in this chapter as criteria.

In this sense, a pivotal language should also be used in different scopes for reuse: 1) The first is for asset producers to specify the information in a common representation, packing and storing it into an arbitrary repository; 2) In a second scope, a client acquires these packages by searching the repository (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 scope, the content of the selected packages is introduced/connected with a specific language, e.g., into a tool chain approach.

The first scope is supported by management operations from repositories. The third scope is supported by DSLs for MDE Settings and tool support for execution of components, chains and reuse operations. In the middle there is a vacuum in the state of the art that hampers the advent of a global reuse scenario: there is no standard representation for description and technicalities associated with MDE Artifacts and their Settings. A first issue is that the lack of requirements makes the procedure for standardization of this information impossible. Therefore, finding these requirements is critical for the progress of research in the area.

We found some, discussed as research gaps for long-term investigations as follows:

For the first scope: For the best of our understanding, in general, important

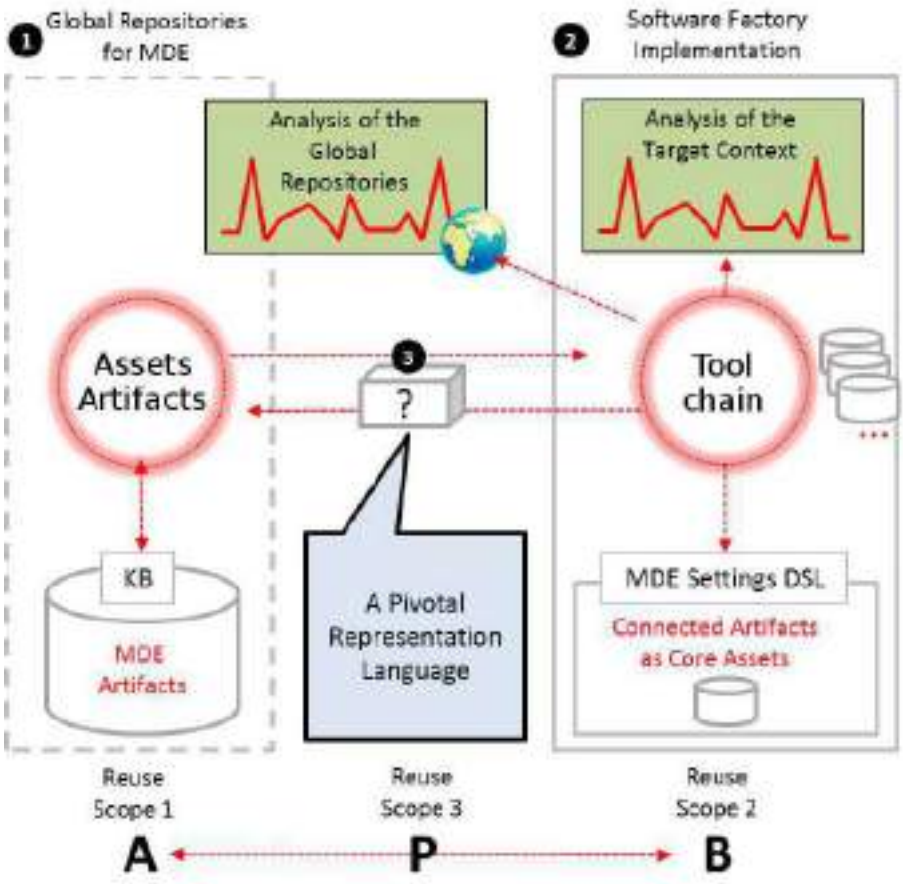


Figure 4.22: Desirable coopetition in MDE as a Service.

information associated with these artifacts is not shared in repositories for MDE, opening window for the following contributions focused in criteria C6: A)Variability points and instructions are relevant, thus structures from these repositories could be incremented; B) The state of the 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) instead of physical files (the focus of repositories)? C) Artifacts should be federated between several possible locations, but the state of the art is limited in this regard, thus open a question on which representations are needed in a KB and/or a pivotal language to manage federated assets and artifacts as currently presented?

For the third scope: The lack of structural features in asset specifications for technical data implies in a manual the effort to publish, search and download these artifacts from repositories: D) We found indispensable a core representation, thus needing an investigation on what is common between Artifacts and DSLs for MDE Settings? E) This lack also needs a manual connection of artifacts in some target representations, thus an open question is whether is possible to do it automatically through a pivotal representation? and F) not everything can be automated, instruc-

tive data and technical data is not well linked in RAS, end-users would benefit from active assistance to connect artifacts with settings, thus it is important to evaluate whether the instructive data represented in assets is useful to assist a manual connection?

For the second scope: In 2007, researchers reported that the lack of critical mass is an issue for the advent of a KB for MDE (FRANCE *et al.*, 2007). This issue is still a research gap (ROCCO *et al.*, 2015). We believe that facilities in tool support can help us to surpass this issue. Thus, open questions include: G) Whether a pivotal language and associated 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, thus an open question is which structural features proposed in the literature is suitable to represent the required data in a pivotal language? and finally J) Which benefits and drawbacks from a common representation for the state of the practice?

Lessons Learned

In this section we revisited conclusions from previous study (BASSO *et al.*, 2016a) and provide lessons learned for the execution of *Acquisition* phase. These lessons are based on assets represented along this thesis, which have been demonstrated in our first experiences (BASSO *et al.*, 2014b) with the intent to simulate an implementation for this phase. In other words, asset consumers have to compare these assets by analyzing their descriptive data. So our lessons are tacked based on previous analysis of criteria, in our previous ad-hoc strategy to represent assets and in our effort to define common representation in RAS++.

As suggests the references associated with criteria C6, artifacts represented in assets are hybrid. Asset representations adopted by each platform discussed in each reference are hybrid too. This means that structural features for describing assets are dependent from artifact domains. For example, SECO platforms consider some structural features non considered by MDE repositories and vice-verse. This means that a common representation language must be flexible to represent these structural features too.

Due to these differences, Section 4.2.1 did not provide specific details for representations in each domain. Instead, we provided general properties that can be used in any context, but we defined criteria C6 on the specificity of MDE to be explored in next RAS++ metaclasses. Figure 4.23 show an example of these metaclasses for representation of data following the criteria C6. These metaclasses are depicted latter in Section 4.3. Lines 1 and 5 shows two instances of `Repository`, the first repreenting a local repository and the second a global. This means that

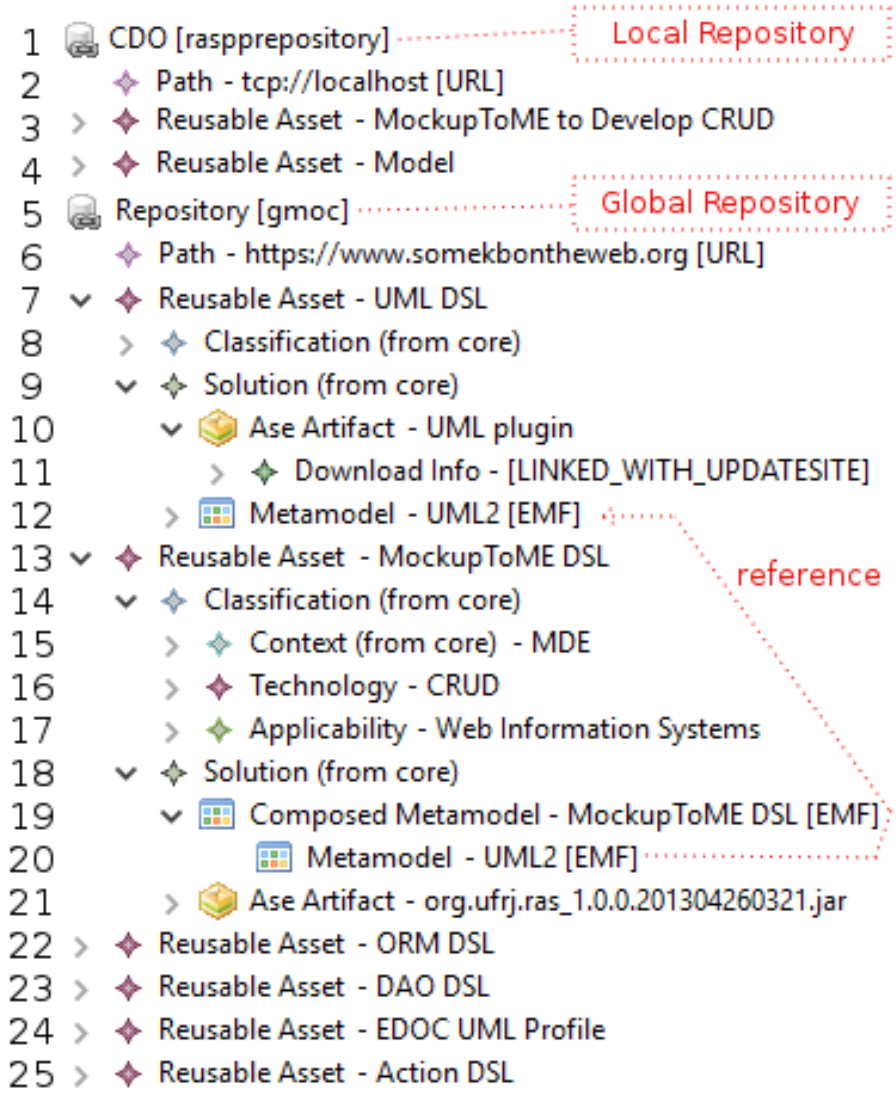


Figure 4.23: Assets representing DSLs

assets in lines 3 and 4 are locally federated while others found in lines 7, 13, 22-25 are federated globally in the GEMOC repository.

Structural features in RAS++ matched the representational criteria suggested by the literature of the area. So far so good, but along these years, we proposed increments in properties and metaclasses for RAS++ to detail classification structures shown in Figure 4.20, in new metaclasses that support classification. Example of these metaclasses are `Technology` and `Applicability` shown in Figure 4.23, lines 16 and 17. This smell bad because each repository proposes their own structural features in terms of descriptor groups for classifications. As a negative result of introducing in RAS++ metaclasses for classification, we ended-up in year 2016 with metamodel composed by more than 80 metaclasses for specific classifications.

Since a big metamodel should be avoided (STRITTMATTER and HEINRICH, 2016), we decided to avoid the introduction of specific classifications as metaclasses

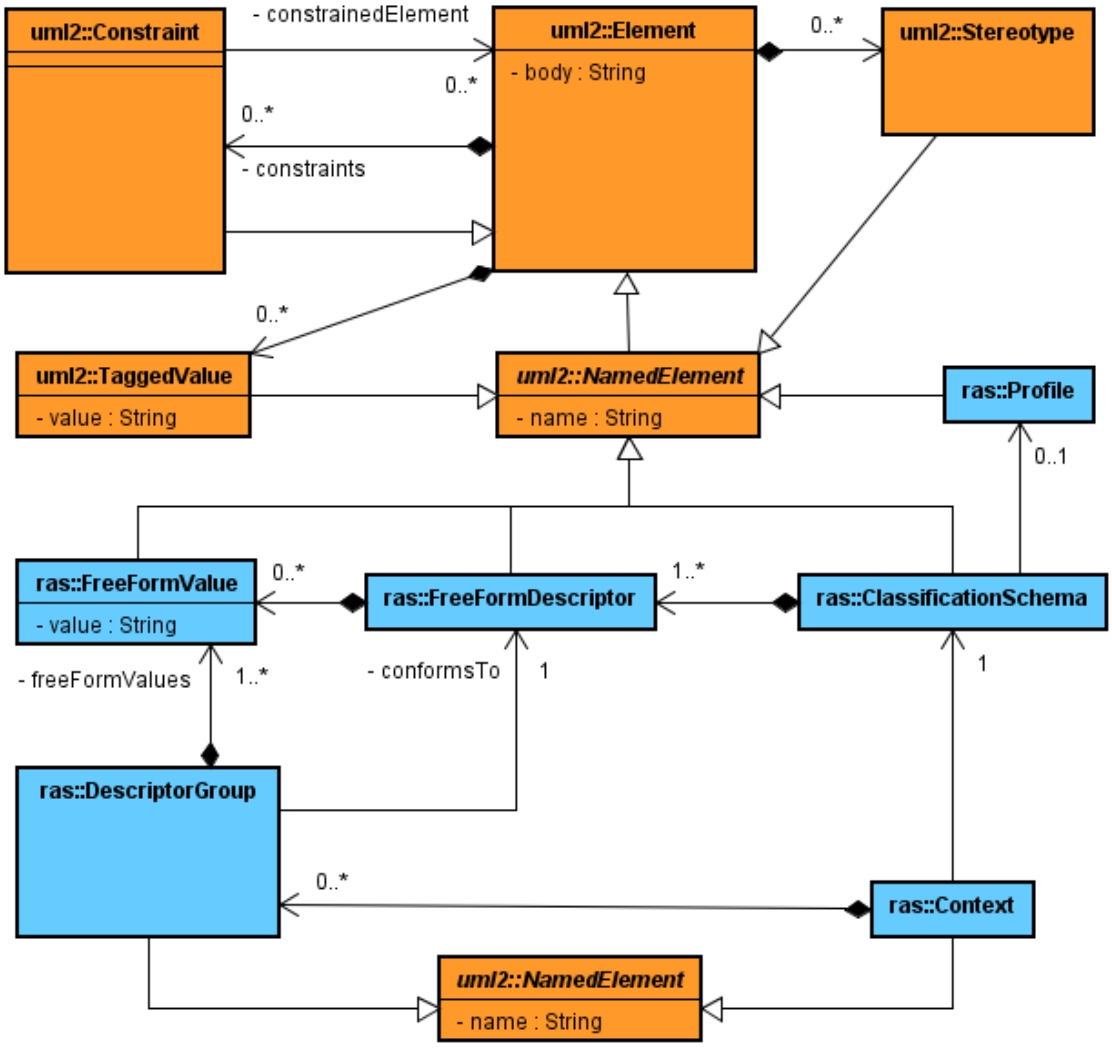


Figure 4.24: RAS++ metaclasses in support for classification schema.

in RAS++. This created an issue to fit RAS++ into the criteria C3 "Well-formedness Features". The next section show how we fixed this issue.

4.2.3 RAS++ Metamodel and Exemplification

In order to demonstrate the flexibility of RAS++ to represent assets conform to the presented criteria, in this section we revisited the RAS++ metamodel, presenting metaclasses not yet discussed. Figure 4.24 shows metaclasses available for representation of structured descriptive data (structural features). At the top-part, metaclasses extracted from UML2 include: `Element`, `NamedElement`, `Constraint`, `TaggedValue` and `Stereotype`. At the bottom-part, metaclasses extracted from RAS include: `Profile`, `ClassificationSchema`, `FreeFormDescriptor`, `FreeFormValue`, `Context` and `DescriptorGroup`.

As exemplified in the Specification phase, descriptor groups allows the represen-

tation of descriptive properties in classifications of assets. Descriptor groups are composed by free form values, which represent leaf values for classifications. The asset provider can use any value he/she wants in free form values. However, some of the structural features presented through criteria need a uniform representation (correct). For example, cost properties (AXELSSON *et al.*, 2014) must consider one of the following values: development cost, product cost, operating cost and information cost. Any different value reduces the quality of the asset representation.

At a first glance, this seems to be a borderline issue. However, since RAS++ aims at pivoting repository representations, a wrong representation imply in violation of well-formedness rules implemented by repositories. Thus, well-formedness rules must be represented in assets.

The novelty is that RAS++ allows the representation of these rules, therefore matching the criteria C3.2 and C3.4. In this sense, a descriptor group (instance of `DescriptorGroup`) must be in conformity with an instance of `FreeFormDescriptor`. In a short definition, a descriptor group follows the rules from a free form descriptor. In the same way, instances of `Context` follows the rules defined in a classification schema (instances of `ClassificationSchema`).

Through associations between these metaclasses, it is possible to formally validate structural features for description of assets. Moreover, since any RAS++ metaclass extends at least `Element`, all the objects/elements may associate instances of `Constraint`, `TaggedValue` and `Stereotype`. This allows the representation of OCL rules as instances of `Constraint` inside each asset element. This is an important characteristic from RAS++, not allowed in AMS and RAS, thus allowing a program to check whether an asset representation is valid.

Figure 4.25 presents the same example used for the Specification phase. The goal is to validate the representation resultant in Figure 4.25.A, which describes an asset using free form values, to be published in a component repository (Repo01) that uses the structural features from (PALUDO *et al.*, 2011). Thus, this asset is structured conforms to Repo01.

Figure 4.25.B shows our representation of structural features from Repo01. In line 1 it is represented a classification schema called "Software Component", which is latter associated with the instance of `Context` shown in Figure 4.25.A, line 1. This context, therefore, follows the rules from the classification schema.

Figure 4.25.A, line 5, shows an instance of `DescriptorGroup` titled as "type" that is associated with an instance of `FreeFormDescriptor` shown in Figure 4.25.B, line 2. This group follows the rules defined on the associated descriptor: 1) the possible values are introduced in Figure 4.25.B, lines 4-7 and; 2) structural/representation rules are defined in line 3 with an instance of `Constraint`.

Figure 4.25.C shows properties from this constraint: at the top-part an OCL rule

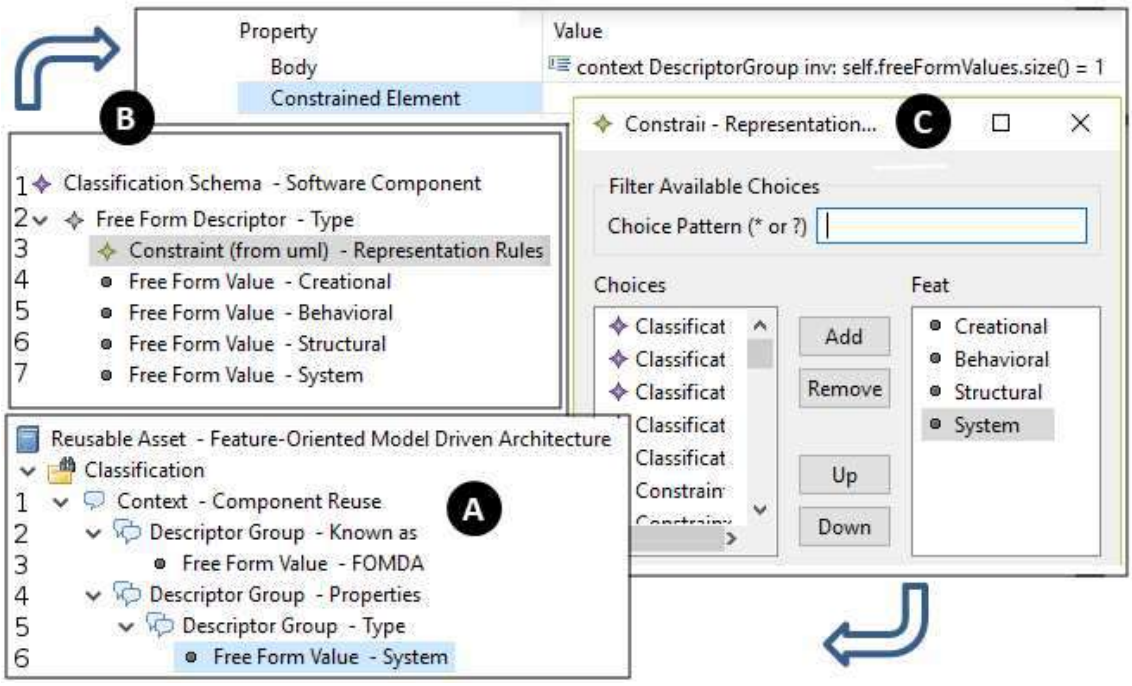


Figure 4.25: UML constraints in descriptive representations.

is expressed as "context DescriptorGroup inv: self.freeFormValues() = 1". This rule can be checked by any OCL tool and semantically means that the associated descriptor group ("Type") must have one and only one instance of `FreeFormValue`. Possible values for the descriptor group "Type" are "Creational", "Behavioural", "Structural" and "System". They are inserted in the association called "Constrained Element" with the help of the illustrated the dialog box.

Figure 4.26 demonstrates that RAS++ is well tailored for representation of properties from our criteria qualified data. Elements from criteria "C1 - Presentation Features" and "C2 - Usage Features" are used and organized as compositions of classification schemes, free form descriptors and free form values. Thus, this example shows how flexible RAS++ is to structure descriptive data from different classifications (A and B) found in different repositories.

4.2.4 Final Remarks

In this section, it was presented an analysis of the state of the art in asset representation. specifically, we analyzed proposals representing decision making data in assets, the so called qualified data. These proposals do not follow the same structural features in descriptor groups. This is natural, since they are built considering specific application and business domains. However, this scenario presents hybrid asset representations.

Hybrid asset representations make harder the implementation of coopetition in

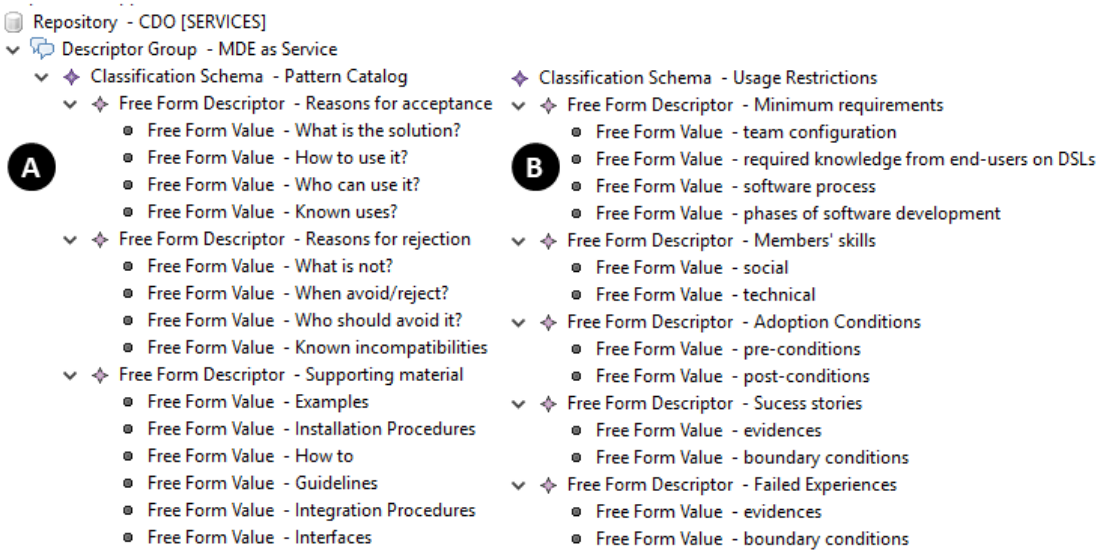


Figure 4.26: Classification groups for criteria C1 and C2.

these heterogeneous scenarios. The integration of these scenarios is our long term goal. For instance, we observed that these hybrid representations of structural features can be represented in RAS++ as well. Our study suggests that, with a reduced number of metaclasses, RAS++ maps structural features adopted by all the discussed/surveyed proposals, thus a feasible language to bridge repositories such as ReMoDD, GEMOC and SEMAT.

Contrarily to our previous observations (BASSO *et al.*, 2016c), this new analysis and demonstration suggest that new extensions in RAS++ for domain-specific classifications are no longer necessary. For this reason, we did not introduce new extensions in RAS++ for the *Acquisition* phase. On the contrary, we removed a set of metaclasses introduced before (BASSO *et al.*, 2013b, 2014b) in support for classifications on the MDE domain. Thus, our contribution is twofold: an analysis of criteria for qualified assets and a cleaning in RAS++.

We also demonstrated alternatives for validating structural features for description of assets. Through a small set of metaclasses, asset providers are allowed to represent constraints in descriptor groups. As result, it is possible to map rules for validation of representations of free form values, which allow constructions for classification of any nature. This is not fully allowed in RAS, neither partially in AMS, thus a contribution from RAS++.

We have demonstrated the use of OCL rules represented in design-time of assets. This possibility makes RAS++ well tailored to represent structural features or descriptor groups independently from rules from a repository.

Finally, we conclude that RAS++ can represent all the introduced criteria, except for one: C6. Criteria 6 demands representations for properties from MDE Artifacts and Settings. Thus, the next chapter details new metaclasses in support

for technicalities required by C6, such as tool chain.

4.3 Asset Transformation

The conception of a common representation language in a technical-level is not a simple task. First of all, it requires technical knowledge and experience from the researcher to determine common abstractions that can be represented with unambiguous properties. In this process, we had to consider properties from different representation languages for assets, MDE Artifacts and Settings, which are diverse, hybrid, ambiguous and not yet properly mapped by the literature of the area. In order to support future implementations in MDE as a Service through this language, several meetings were carried out with research collaborators to find out what is feasible to introduce in RAS++. Our aim is to report on this effort, deriving the metamodel we proposed for representation in asset some technicalities associated with MDE.

In this sense, it is important to consider properties from modern toolboxes for MDE, which are built on domain specific languages (CUADRADO *et al.*, 2014; LEMOS and MASIERO, 2011; SYRIANI *et al.*, 2015) for representation of MDE Artifacts and Settings (HEBIG, 2014) (MDE components). Due to the introduction of new design languages for systems and processes (JOUAULT *et al.*, 2010), as well as to the evolution in technologies used in underlying implementation frameworks (HEBIG and BENDRAOU, 2014), components such as MDE Artifacts and their Settings are often analyzed to fit to the context of a target software project (BASSO *et al.*, 2016d). A decision about the use of these components is therefore dependent from the needs of stakeholder working in companies/contexts (WHITTLE *et al.*, 2015), i.e., depends from an asset consumer (AXELSSON *et al.*, 2014). This means that in the last phase, technical properties from MDE Artifacts and Settings should be considered. Therefore, in order to support the *Transformation* phase, it was introduced new RAS++ extensions.

This section is organized as follows. Section 4.3.1 summarizes our mapping study by which we extracted common properties from tool chain approaches. Section 4.3.2 presents other extensions for the RAS++ metamodel. Finally, Section 4.3.3 demonstrates the use of RAS++ in the same problem domain discussed along the previous chapters, and Section 4.3.4 encloses this chapter with final remarks.

4.3.1 Mapping Study

A limitation in the literature is the lack of studies exploring the state-of-the-art in MDE Settings (BASSO *et al.*, 2017a). For instance, several tools have been

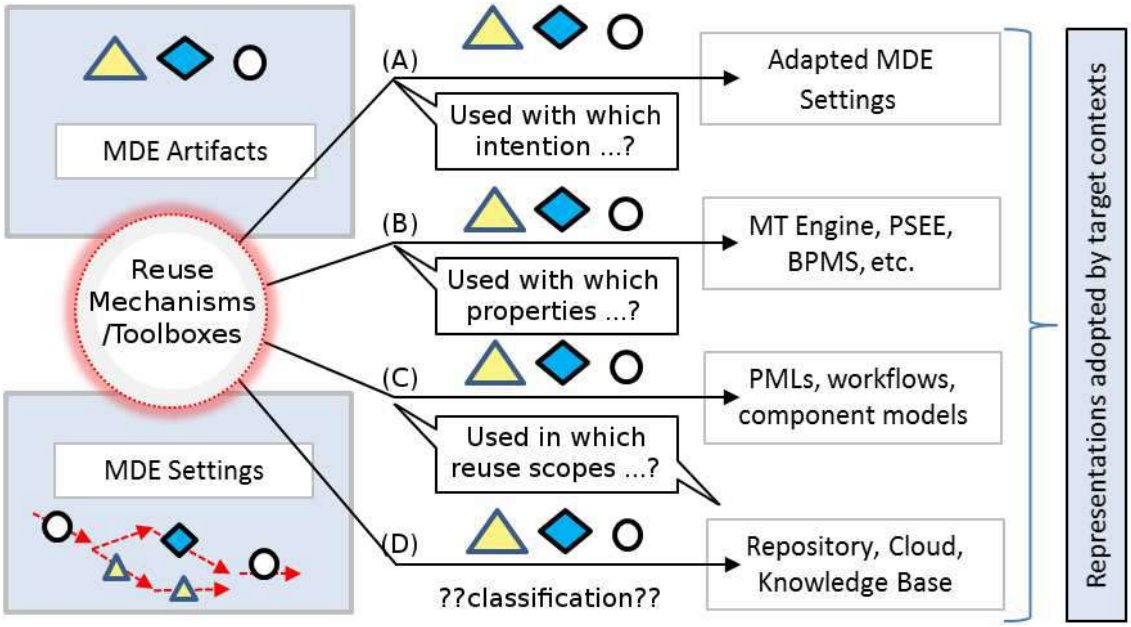


Figure 4.27: Elements for MDE Settings considered in this mapping study

proposed as a way to help software engineers to introduce MDE artifacts in target contexts independently from a model transformation language. Such tools are built based on many concepts introduced in software engineering ranging features such as integration, services, orchestration, consistency, semantics and execution of artifacts/resources associated with models and transformations.

Goal

Our goal is to extract relevant properties from approaches for tool chain for inclusion in RAS++. In this sense, the following question motivates this mapping study: **Q5.5: Which are the properties required in RAS++ in the context of MDE as a Service for tool chain representation?**

Research Method

In order to reach the MDE Settings intents and their properties, we conducted a structured review of type mapping study (PETERSEN *et al.*, 2008). As illustrates Figure 4.27, in this study we are investigating which are the representations available in some representation languages that seamlessly link tasks, tool support, components and DSLs. The overall study is presented in a technical report[7].

[7]MDE Settings Intents and Their Properties - `prisma.cos.ufrj.br/wct/ms04.pdf`

Analysis

This section reports on our answer to **Q6: Which are the properties required in RAS++ in the context of MDE as a Service for tool chain representation?**

General properties include following component taxonomy that is not adequate for representation through descriptor groups and classifications in RAS++:

Semantics for MDE, such as associate descriptive information with transformations.

Chain of endogenous model transformations, which means the usage of the same metamodel associated with models generated through refinements;

Chain of exogenous model transformations, which means the usage of more than one metamodel associated with models generated through model-to-model transformations;

Chain of built-in model transformations characterizes approaches that enable insertions, in run-time of a tool chain, of small programs (algorithms/model operations).

Structural features for components include:

Component properties such as Input and Output (IO) parameters, associated scripts or programs for execution of model-based operations.

Atomic component is indivisible, which means that a component cannot be expanded into sub-components. A component is also called as task or activity in tool chain approaches that consider integrations with process modeling languages. This is an ambiguity in the area. Thus, in RAS++ they all are called as component.

Composite component is the opposite. It is also defined as sub-process, sub-activity and sub-routine in tool chain approaches. In RAS++ these properties are called composite components.

Component fragment is an artifact derived from fragmentation, usually performed with the assistance of a software product line toolbox.

Approaches for MDE Settings represent execution flow using different approaches. We selected the following execution-flow properties:

Execution Index, used in component-based approaches, represents the sequence of executions for sub-components inside another component.

Transitions and workflows are used in process-based approaches. Workflow representation is included in our previous works in RAS++ (BASSO *et al.*, 2014b), but it is out of the scope in this thesis.

Parallelism or commutativity is the representation that two or more components can be executed in paralell.

Tool chains are represented as a sequence of inputs and output in parameter matching approaches. This way, **IO Typing** characterizes properties such as

parameters (see metaclasses "BasicDataType" and "ModelType" from the Wires* metamodel) (RIVERA *et al.*, 2009) or the metaclass "Binding" in the Bentõ DSL (CUADRADO *et al.*, 2014). Model typing is also called as model subtyping (GUY *et al.*, 2012)) and allows to apply techniques for consistency check in IO bindings used in co-evolution of model, metamodels and transformations.

In the following, we classify approaches for parameter matching whose properties are considered for inclusion in RAS++.

Datatype classifies approaches that represent in IO parameters from components primitive types such as String, Date, etc. Wagelaar considered black-box compositions of model transformations whose parameters are checked considering a native data-type (i.e., the Java data-type) (WAGELAAR, 2006).

Metatype is the class specified in a meta-model and it is manipulated by model transformation rules (AZANZA *et al.*, 2010). In other words, model transformation tasks (also called as operations and rules) can be orchestrated according to IO parameters that owns information about meta-type. This means that the check of validity in bindings must consider types that inherit from other types in a metamodel.

Interface is a variation from Datatype. Rose et al. considered the bind between IO transformation parameters by comparing two definitions of an UML interface (ROSE *et al.*, 2013)

Artifact Typing characterizes approaches that add typing for general artifacts (e.g., define an artifact as a library, models, test case, APIs, tutorials, etc) (HONG-MIN *et al.*, 2009; VIGNAGA *et al.*, 2013; ZHANG and MOLLER-PEDERSEN, 2014).

Depending on the IO Typing, MDE Artifacts can be assigned to parameters through bindings. Binding is the ability to connect artifacts into a MDE Setting. For example, assigning values to IO parameters from components such as an input model. The following properties are found in binging and are included in RAS++:

Meta definitions include metadata conceptions for model, metamodel, and metametamodels. This is a requirement for any approach for integration in tool chain through MDE Settings.

Integration Filter and Ports control the heterogeneity in tool support with toolboxes supporting model-driven tool integration (BIEHL *et al.*, 2014). Integration filter is the equivalent to a transformation component and ports are equivalent to transformation parameters. Thus, we removed this ambiguity, keeping in RAS++ the name transformation component and parameter as a common definition.

Serializations are adopted by toolboxes to export models in different formats.

Interface specification is the representation of artifacts in support for web services.

Finally, **Toolboxes** are abstractions for systems, no matter which system. They include interface specifications and serialization formats, are composed by components and by all the aforementioned structural features and properties. Therefore, a representation of toolbox considers low and high-level properties.

4.3.2 RAS++ Metamodel

Based on properties found in integration approaches for MDE Artifacts and Settings, we proposed new metaclasses in RAS++. This section presents what we found as common between MDE Artifacts and Settings.

MDE Settings

Figure 4.28 presents part of the RAS++ metamodel with the possible types that can be used in a model transformation parameter (instance of metaclass `TransformationIO` shown in Figure 4.29). Instances of `Repository` and `Asset` own instances of the UML metaclass `Type`. Instances of `Type` are used as references in instances of the UML metaclasses `Property` and `Parameter`, shown in Figure 4.29. Therefore, the association between `Property` and `Type` allows to represent in RAS++ the same semantics from the dependency stereotyped with «conforms to» in the FOMDA DSL, associating an instance of `Metadefinition`, or data types that are usually represented in parameters of model transformation rules or operations (BASSO *et al.*, 2014e).

We found in DSL for MDE Settings some similar concepts such as task/component and sub-process/sub-component in approaches for MTC, CMMT and MDE-SDP. These elements are also of interest for a common representation language, but they need an unification in RAS++. Our proposal is a unified concept by artifact for model transformation of type "atomic" and "composed". Transformations are instances of `MDEArtifact`. Atomic components are represented in RAS++ as an instance from the metaclass `AtomicTransformationArtifact`.

Model transformation chains, processes and composed components are represented with instances of `ComposedTransformationArtifact` shown in Figure 4.29. The literature states that, in compositions of model transformations, it is important to inform if the nested artifacts are sequential. Sequential elements must be integrated to MDE Settings following a sequence defined by the property `index`. Another important concept introduced by MDE Settings is the commutativity, i.e., when sub-artifacts are commutative for the implementation of the parent artifact. This information is also important for the role "Transformation", when these artifacts are integrated to DSLs for MDE Settings.

Representations for software processes, i.e., in DSLs for MDE-SDP and MDE-PP,

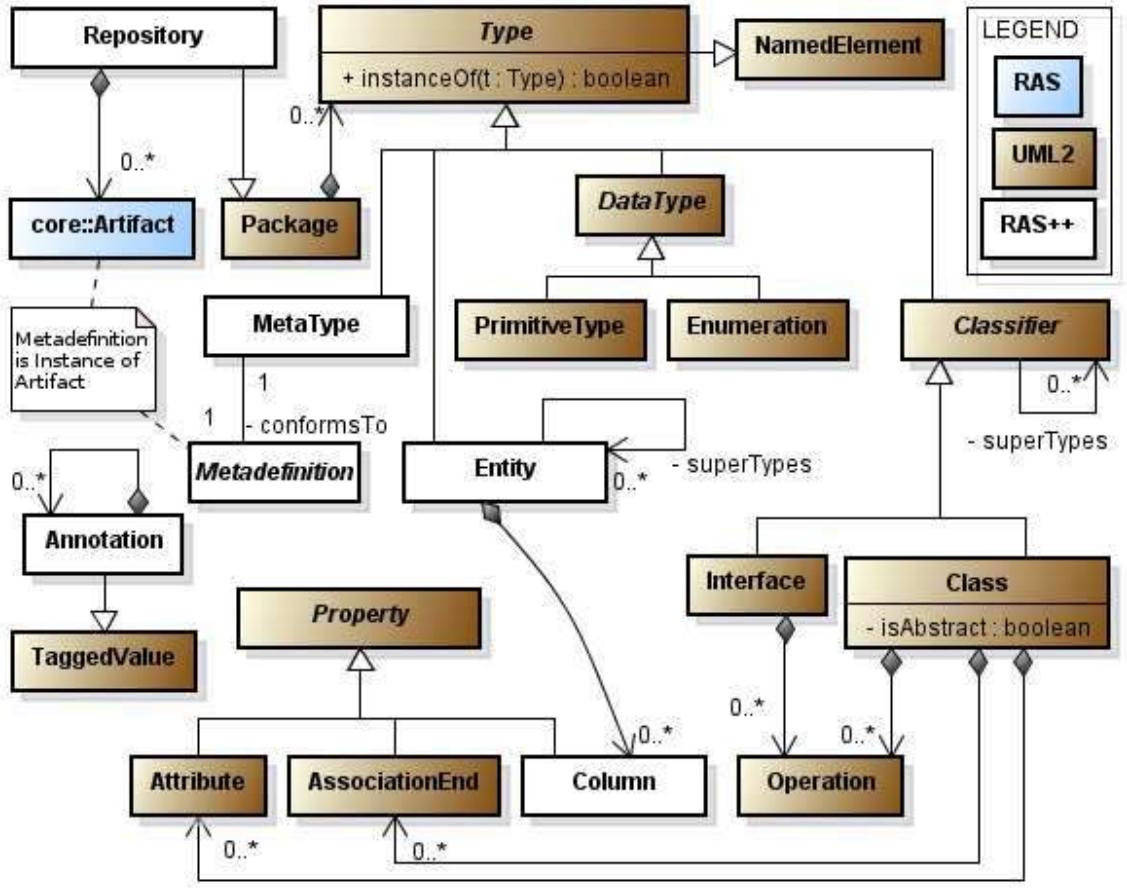


Figure 4.28: Metaclasses for typing MDE Parameter.

adopt concepts for artifacts. This concept is similar to those adopted in DSLs for MTC and HOT (pin and parameter), for CMMT (interface) and for MDI (port). The concept of artifacts from DSLs for MDE-SDP and MDE-PP is very different from the concepts of artifacts from assets. This needs a uniformity in RAS++ also in terms of conceptualization. Artifacts in RAS, AMS and RAS++ must not be confused with the aforementioned concepts. Thus, we proposed a unique representation for this concept through the metaclass `TransformationParameter`, keeping the concept of `Artifact` as found in pivotal representation languages.

We also added relationships to metaclasses related with semantics for MDE artifacts. For example, by representing instances of `ContextRef` in MDE artifacts, it is possible to reference a context described by the asset. References for the semantics for classification shown in Figure 4.29 are represented in associations with metaclasses `Context` and `ContextRef`. This allows to provide rich descriptive information directly in instances of `MDEArtifact` and also through associations between `AtomicTransformationArtifact` with `ContextRef`. Moreover, an instance of `Artifact` allows the representation of instances of `ArtifactContext`. Finally, semantics for MDA views are expressed in properties from the metaclass `AtomicTransformationArtifact`, such as the property `intention` whose possible values are defined in the

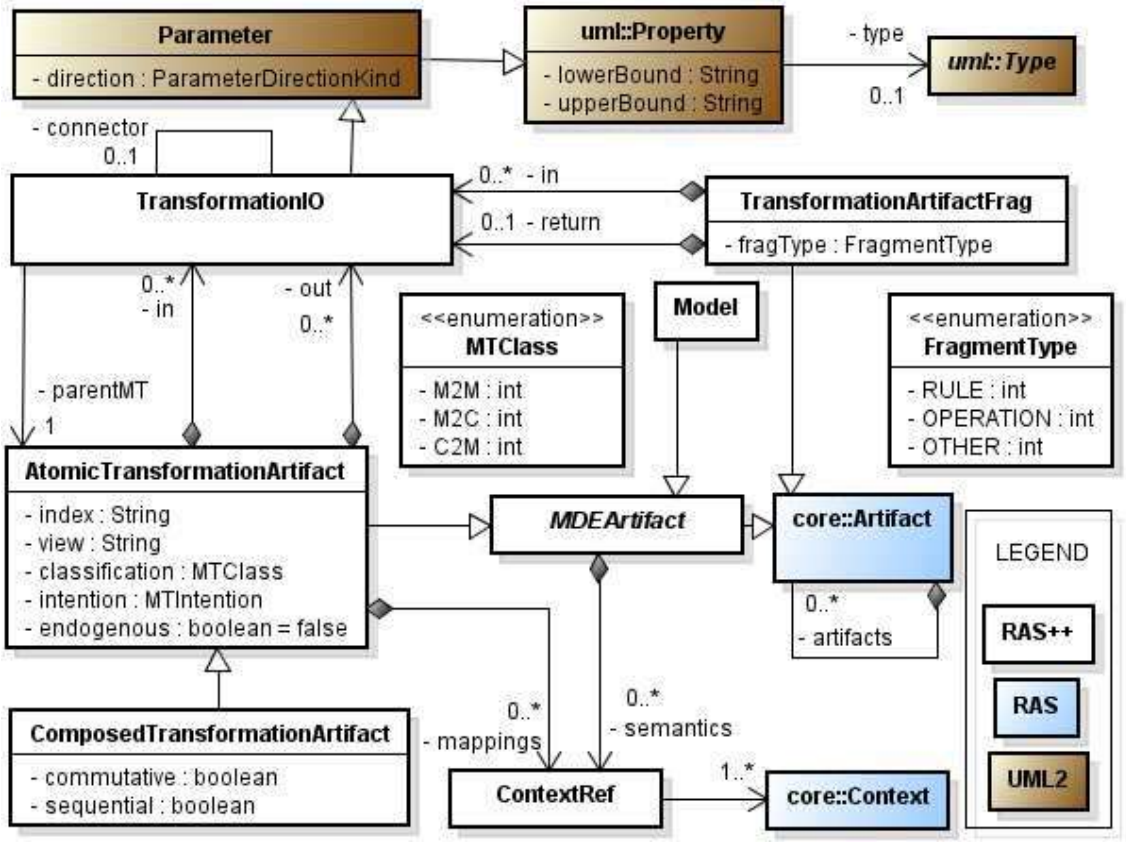


Figure 4.29: Metaclasses for model transformation components.

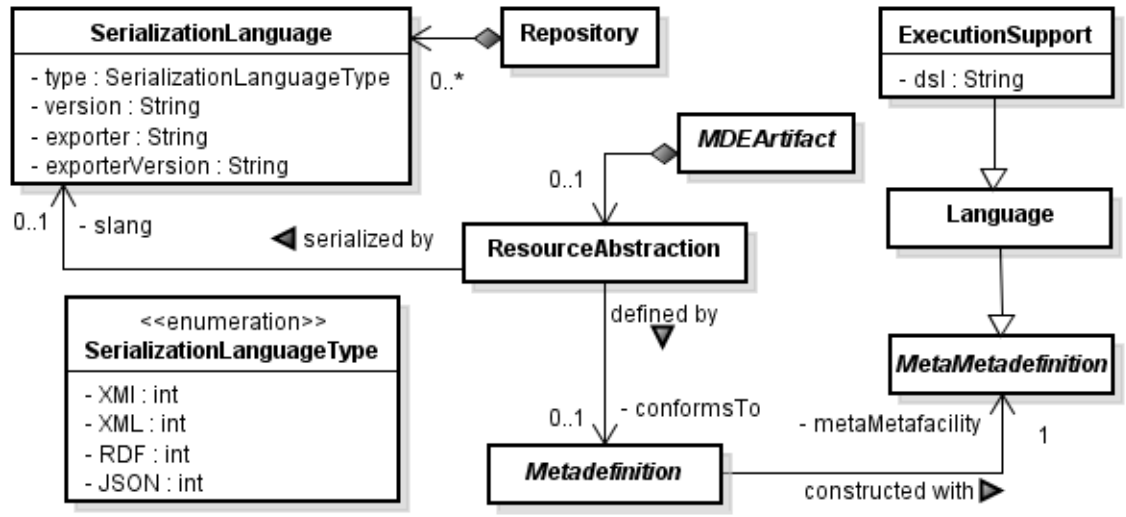


Figure 4.30: RAS++ metamodel to specify abstractions for MDE artifacts.

literature as standard taxonomy for model transformation intentions (LÚCIO *et al.*, 2014).

MDE Artifacts and Types

As shown in Figure 4.30, the metaclasses `ModelType` and `MDEArtifact` are associated with the metaclass `ResourceAbstraction`. This allows the representation of details about artifacts and model types as common representations in DSLs for MDE Settings. Instances of `ResourceAbstraction` are used to represent: 1) the metamodel associated with the resource through an association with `Metadefinition`; 2) the serialization language (e.g., XMI, XML, etc) used to import and export the resource through an instance of `SerializationLanguage`; and 3) the `Language` associated with the artifact, such as for software artifacts of type source code the language Java or for components for web services the language SOAP. The first two associations are specific for MDE Artifacts, while the third association is for software components in general.

Model typing is a good solution to detect incompatibilities in the second reuse scope. However, we also experienced issues regarding the serialization of XMI, which is exported differently by some UML design tools. This way, we introduced in the metaclass `SerializationLanguage` the properties `version`, `exporter` and `exporterVersion`. This information is structured and descriptive, used by reusers for decision making, thus not precessed by programs such as model checkers.

Another element introduced is for detail in which format is the model represented. It is acknowledged that in MDE the use of XMI is common. However, new proposals for serialization are adopting also representations in RDF and JSON. This way, we introduced in the metaclass `SerializationLanguage` the classification of the type of serialization, as literals from the enumeration `SerializationLanguageType`. Through the serialization language it is possible to acknowledge this important requirement for the execution of a model transformation engine. Thus, if necessary, the software engineer can search and download a model transformation with the intent of serializing the data (LÚCIO *et al.*, 2014) for a format supported by the adopted engine.

Besides model typing, which is important for the representation of MDE Settings, artifacts such as models and transformation components can associate instances of `ResourceAbstraction`. Figure 4.30 shows that instances of `ResourceAbstraction` associate one instance of `Metadefinition`, which associates one instance of `MetaMetadefinition`. These relationships are important to represent with which model compiler an artifact is built. For example, some model transformation components of type model-to-model are constructed with model transformation languages such as ATL or ETL. These are model compilers constructed with metamodels that are built with meta-meta-facilities such as language generators including EMF and grammars such as Abstract Syntax Tree (AST). In special for model transformation

components that need chains IO, the information for resource abstraction allows the correct use of model types in parameters such as to constrain the use of `MetaType` or `DataType`. This is discussed in Section 4.3.2 with OCL invariants.

Metadefinition and MetaMetadefinition

The options for representation of resource abstractions are shown in Figure 4.31 in a metamodel-part for MDE Artifacts associated with the concepts of meta-definition and meta-meta-definition. The common use in MDE Settings is to represent an instance of `Metamodel` associated with an instance of `MetamodelGenerator`, used to apply validations of consistency in parameter matching (ETIEN *et al.*, 2012) and recommendations (BASCIANI *et al.*, 2014b). However, recent works are also including models in conformity with JSON (IZQUIERDO and CABOT, 2013) and also ad-hoc DSLs whose models are in the format of XML, such as graphical user interfaces and forms represent with tools such as the Graphical Designer from Netbeans IDE[8]. Besides, in our motivating scenario, some DSLs are built on top of others, requiring the representation of artifacts as composition of DSLs.

Our proposal is to organize these concepts common from DSLs for MDE Settings in MDE artifacts. Figure 4.31 provides details about metaclass `Metadefinition`. This is a type of MDE Artifact that provides meta definitions about DSLs. Implementations include: 1) `Native`, is the default option (does not need explicit representation) and classifies MDE Artifacts that are built without any support for meta-modeling, such as model transformation components developed with Java; 2) `AdHocDSL`, which represents design languages constructed without the support for meta-modeling, but differently from the first option, are represented in some way in a class diagram or Entity-Relationship (`ER`); 3) `Grammar` that classifies design languages constructed based on Abstract Syntax Tree (AST) resulting in textual DSLs; 4) `XMLSchema`, which classifies some DSLs based on XML such as UsiXML[9]; 5) `Metamodel` that classifies the most common DSLs; 6) `UMLProfile`, which extends `Metamodel` to define conservative extensions; and 7) `ComposedMetamodel` that references one or more metamodels developed/generated independently, such as EMF-based metamodels that extends other ecore files in a heavy-weight approach for extension.

Each instance of `Metadefinition` references one instance of `MetaMetadefinition`. The dependencies illustrated in Figure 4.31 indicates which instances must be associated. Instances of `MetaMetadefinition` include: 1) `Language`, which is any language that can be associated with instances of `Native` Metadefinition; 2) `DTD`, which is used in meta definitions based on `XMLSchema`; 3) `ER` is a way that some ad-hoc DSLs have been created; and 4) Instances of `MetaMetamodel`, which

[8]http://www.netbeans.org/
[9]UsiXML-<http://www.usixml.org/>

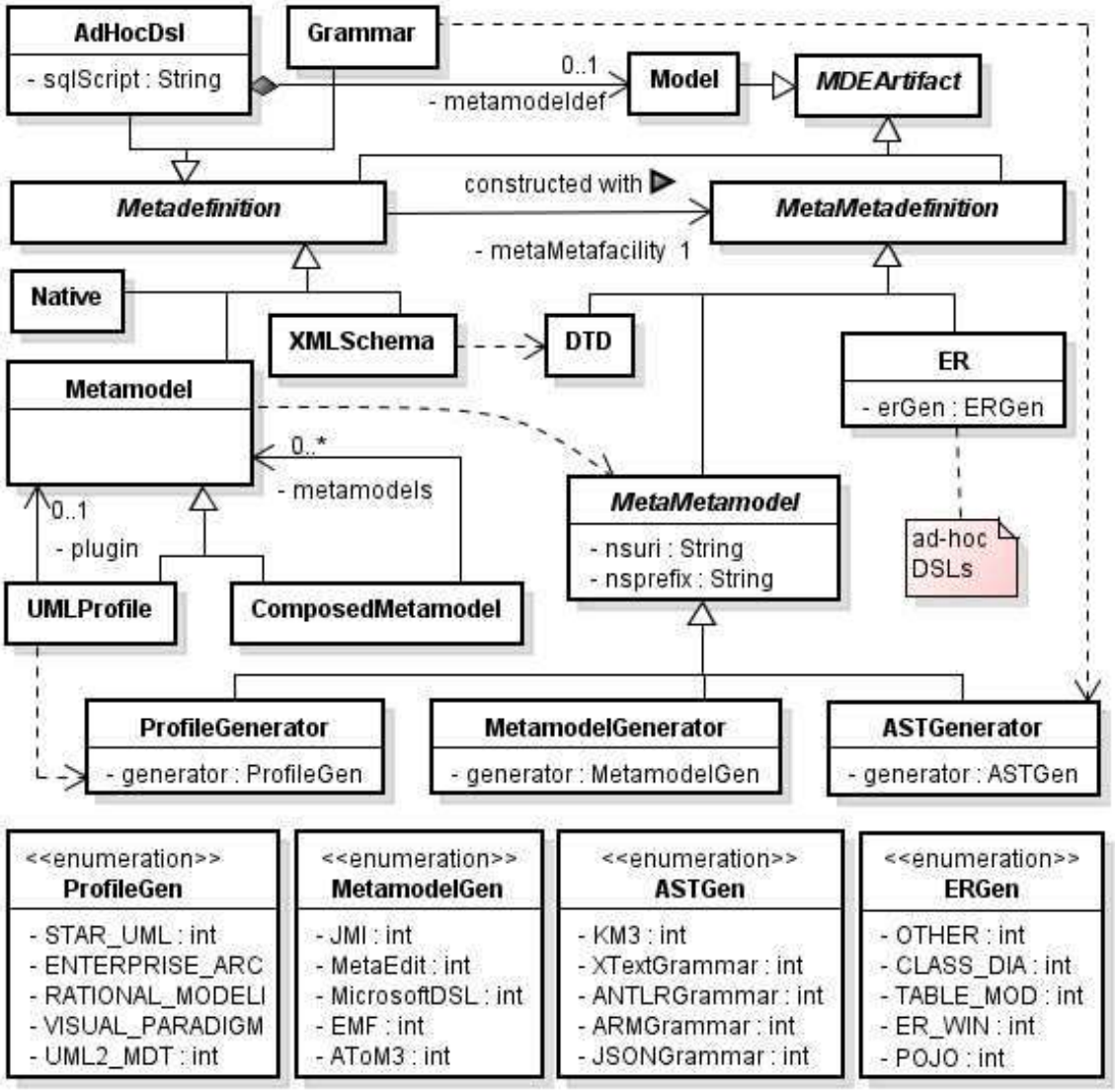


Figure 4.31: RAS++ meta-definition and meta-meta-definition.

are constructed from principles of meta-meta modeling.

Instances of `MetaMetamodel` are more restrict in the literature than those for Metadefinition. In other words, the number of options for tools for meta-meta-modeling is lower than the number for meta-modeling. This way, they can be represented with enumerations shown in the bottom-part of Figure 4.31. Instances of `MetaMetamodel` include: 1) `ProfileGenerator`, which includes UML design tools (some are described in the enumeration `ProfileGen`); 2) `MetamodelGenerator` that includes tools for generation of DSLs described in the enumeration `MetamodelGen` such as EMF and MetaEdit; and 3) Abstract syntax tree generators as instances of `ASTGenerator`, which can be performed with the tools listed in the enumeration `ASTGen`.

```
1  import ecore : './raspp.ecore'
2  import ecore : './uml2.ecore'
3
4  package rasppmde
5
6  context AtomicTransformationArtifact
7    inv: self.artifacts.oclIsKindOf(TransformationArtifactFrag)
8  context TransformationArtifactFrag
9    inv: self.artifacts.oclIsKindOf(TransformationArtifactFrag)
10 context ComposedTransformationArtifact
11   inv:  self.artifacts.oclIsKindOf(AtomicTransformationArtifact)
12 context XMLSchema inv: self.metaMetafacility.oclIsKindOf(DTD)
13 context Metamodel
14   inv: self.metaMetafacility.oclIsKindOf(MetaMetamodel)
15 context Grammar
16   inv: self.metaMetafacility.oclIsKindOf(ASTGenerator)
17 context UMLProfile
18   inv: self.metaMetafacility.oclIsKindOf(ProfileGenerator)
19 context AdHocDsl
20   inv: self.metaMetafacility.oclIsKindOf(ER)
21 context Native
22   inv: self.metaMetafacility.oclIsKindOf(Language)
23 context TransformationIO
24    inv:
25     if(self.parentMT.oclIsKindOf(AtomicTransformationArtifact))
26       then self.type.oclIsKindOf(MetaType)
27     else not self.type.oclIsKindOf(MetaType)
28     endif
29 context TransformationIO
30    inv: self.connector <> self
31       and self.parentMT <> self.connector.parentMT
32       and self.direction <> self.connector.direction
```

Figure 4.32: OCL invariants applied in MDE Artifacts.

OCL Invariants

Artifacts in the standard RAS can be composed by other artifacts. This is good because model transformation chains are artifacts composed by transformation artifacts. However, atomic artifacts cannot be composed by other transformation artifacts, allowing inconsistencies in representations. In order to ensure that only valid compositions are possible for MDE Artifacts and Settings, Figure 4.32 presents OCL. Thus, through RAS++ and OCL invariants, it is possible to bridge the information found in MDE Artifacts and Settings with a more appropriate pivotal language than RAS.

From line 6 to 11, invariants refer to instances of `Artifact` from model transformation components shown in Figure 4.29: a) Line 6, ensures that instances of `AtomicTransformationArtifact` are composed only by instances of `Transformation-`

ArtifactFrag; b) Line 8, ensures that instances of TransformationArtifactFrag are not composed by other artifacts. Instances of Model have the same invariant; and c) Line 10, ensures that instances of ComposedTransformationArtifact are composed only by instances of AtomicTransformationArtifact.

From line 12 to 22, invariants are specific to constraint compositions between instances of Metadefinition and MetaMetadefinition, shown in Figure 4.31, as follows: a) Line 12, ensures that if an instance of XMLSchema is used as meta-facility, then DTD is used as meta-meta-facility; b) Line 13, ensures that instances of Metamodel references a MetaMetamodel instance; c) Line 15, ensures that DSLs constructed with a grammar references an instance of ASTGenerator; d) Line 17, ensures that a UML profile makes a reference for the UML tool as a meta-meta-model; e) Line 19, ensures that instances of AdHocDSL make reference for an instance of ER; and f) Line 21, ensures that instances of Native make reference for an instance of Language, such as Java, C++ or other that is used for the development of black-box model transformations.

The invariant in line 23 ensures that types associated with TransformationIO are represented conforming to the level of the artifact in a MDE Setting. This is possible due to the following rule: instances of MetaType are used only with instances of AtomicTransformationArtifact. In the same way, instances different from MetaType are used in IO when the artifacts are instances of ModelTransformationFrag (e.g., rules and operations from transformations).

Line 29 ensures that a connector between instances of TransformationIO does not link IO inconsistently: a) Line 30, ensures that a parameter does not connect to itself; b) Line 31, ensures that connectors are not established between IO from the same artifact; and c) Line 32, ensures the connection between an input and an output, which is based on the property direction from the metaclass Parameter from the short UML metamodel.

Toolbox

As recently suggested as limitations for execution of preliminary phases for tool chain, ZAKHEIM (2017) claims the need for highlighting all the artifacts needed in integration processes. So, the last extensions are designed in support for toolbox representations, as shown in Figure 4.33. A toolbox is an instance of Artifact and is composed by instances of ChannelAdapterSpecification[10]. The metaclass InterfaceSpecification allows to represent adapter components for remote method invocation, usually accessed through native implementations such as Java RMI. The metaclass WDSL is for representation of artifacts of type adapter web service and

[10]EIP - http://www.enterpriseintegrationpatterns.com/patterns/messaging/ChannelAdapter.html

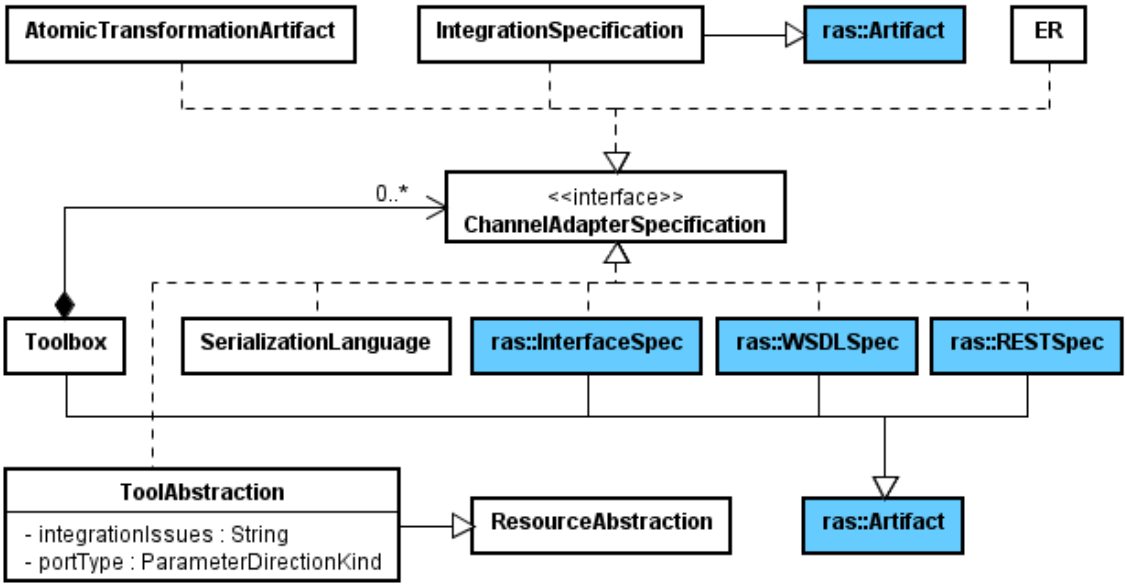


Figure 4.33: RAS++ extensions to support toolbox abstractions.

`RESTService` is used for representation of artifacts for remote method invocation as adapter built on the REST API. These artifacts have been proposed in the literature as components for tools, including some representation in RAS found in Component and WSDL Profiles.

Our novelty is the metaclass `ToolAbstraction`, which allows associations of tool components built on MDE specificity, as well as the insertion of: 1) artifacts with integration specifications (FRANTZ and CORCHUELO, 2012) represented as instances of `IntegrationSpecification`; 2) artifacts as model transformations represented as instances of `AtomicTransformationArtifact`; 3) serialization formats represented as instances of `Serialization` and, 4) entity relationship models represented as instances of `ER`.

This is a mean to promote facilities in preliminary phases for tool chain (ZAKHEIM, 2017), thus highlighting what asset consumers should look at in terms of artifacts when performing tool chain integrations.

4.3.3 Representation of Assets

Figure 4.34 shows a MTC represented in the component-level with the FOMDA DSL. This is our motivating example, used along this chapter to illustrate the application of RAS++ concepts. This example is extracted from MockupToME Method (BASSO *et al.*, 2016d), whose artifacts for transformations assist the development of web information systems.

The model transformation component called “Generate Entities” is of type endogenous. It allows the generation of an adapted UML model. The resultant model

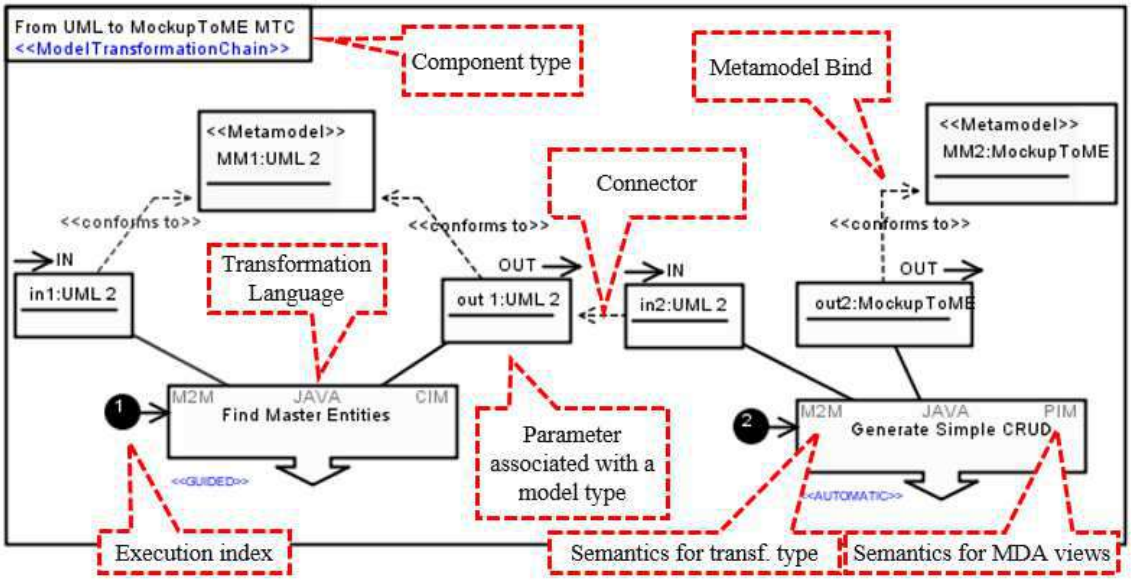


Figure 4.34: Essential settings conform to the FOMDA DSL.

is input for the second component called "Generate Mockup", which is exogenous (endogenous=false). This second component allows the generation of models in conformity with MockupToME DSL (BASSO *et al.*, 2016d). In RAS++ these components are instances of `AtomicTransformationArtifact`.

As a common representation found in the literature, MDE Artifacts owns parameters that are typed. While parameter types are used by software programs to check consistency (ETIEN *et al.*, 2012) and to recommend compositions through searches in repositories (BASCIANI *et al.*, 2014b), other information associated with components is for semantics purposes such as transformation view, classification and intents. Figure 4.34 shows two artifacts stereotyped as «Metamodel» that are used for parameter typing in MDE. For example, the first input parameter called "in1: UML 2" is associated with a "Type" that links to the metatype "MM1: UML2".

The composition of these artifacts classifies a MDE Setting. This setting is represented as an artifact called "From UML to MockupToME MTC", a MTC composed by atomic artifacts. In RAS++, this MTC is instance of `ComposedTransformationArtifact`. These artifacts are organized in sequence, which imply in RAS++ in setting the property `sequential` from an instance of `AtomicTransformationArtifact` for `true`. Moreover, the exemplified composition is not commutative (i.e., `commutative=false`), thus internal components are not substitutable inside the MTC.

As motivated in this chapter, a goal from some researches is to share these artifacts on the web through the globalization of DSLs. Artifacts in conformity with the FOMDA DSL are not adequate to be shared in knowledge bases because they are not in the format of repositories for MDE Artifacts proposed in the literature (e.g.,

MDEForge, GMOC or ReMoDD). Moreover, in order to avoid the combinatorial explosion that would occur in transformations between MDE Settings DSLs, these MDE Artifacts must be represented with a pivotal language. This is demonstrated in the following.

Running Example

Figure 4.35 shows an asset that represents the information from the FOMDA DSL with RAS++. The Asset "MockupToME to Develop CRUD" associates information about a set of model transformation artifacts. The exemplified assets present some artifacts such as metamodels, libraries, model transformation chains and other artifacts such as libraries that are associated with these specifications. We will not demonstrate the representation of semantics for classification and profile, illustrated in lines 2 and 3, because these elements are well explored by our previous works. We focused in representations for MDE Artifacts and Settings, which belong to the `Solution` structure shown in line 4.

Data associated with MDE Artifacts and Settings are represented between lines 8 and 17. Differently, a regular artifact (conforms to the default RAS Profile) is illustrated in Line 5 with the available structure for semantics (classification and artifact typing) shown in lines 6 and 7. The information between lines 8 and 17 is the semantics and syntax for two model transformation components inside a MTC. For example, the component called "Generate Entities" is represented in RAS++ with an instance from the metaclass `AtomicTransformationArtifact` in line 9. The component shown in line 12 is the second component in the sequence (index=2) and, differently from the "Generate Entities", can be adapted (see the variability point in line 13). This semantics for adaptation provided in line 13 is instructive, a useful element for highlighting a need for component integration before use, as exemplified by our previous contribution (BASSO *et al.*, 2014b).

Atomic artifacts shown in lines 9 and 12 own typed parameters. As in FOMDA DSL, parameters shown in lines 10, 11 and 14 link to the metamodel "UML 2", while the parameter shown in line 15 links the MockupToME DSL metamodel. The input parameter shown in line 14 has a connector referencing the output parameter shown in line 11. Finally, semantics for classification are represented in lines 16-17, connecting information from MDE Artifacts and Settings with information needed in repositories for classification.

Figure 4.36 illustrates the representation of data associated with instances of `ResourceAbstraction`. The first asset illustrated between lines 1 and 11 shows two models in lines 5 and 9. Each model is constructed with a different metamodel: M1 is constructed with a UML Profile called ORM Profile and M2 with a composed model called MockupToME DSL. Moreover, M1 is serialized in a specific format shown in

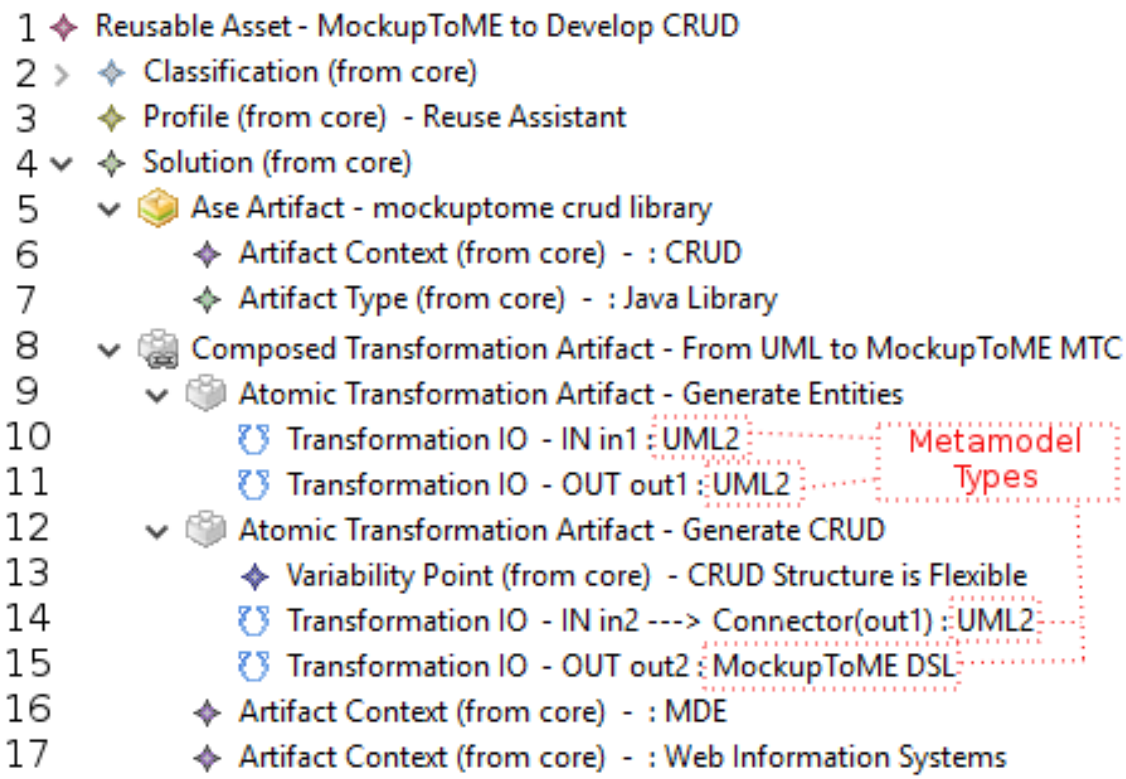


Figure 4.35: Representing MDE Settings

line 8, while M2 is assumed to use the default exporter associated with the corresponding meta-meta-model generator "[EMF]", whose representation is illustrated in Figure 4.37.

Resource abstractions are also used in associations with MDE Settings. The second asset shown in Figure 4.36 complements the previous example by detailing the constructors associated with the artifacts instances of `AtomicTransformationArtifact`. The abstraction associated with the MTC in line 16 links the FOMDA DSL metamodel with the artifact "From UML to MockupToME MTC". The abstraction shown in line 19 indicates that the artifact "Generate CRUD" is built with a native language "JAVA".

This information is used for contextualization purposes in a pivotal representation language. However, it is useful for the first and second scopes for reuse, when applied in MDE Settings DSLs and tools. For example, the exemplified setting can be used to check consistency in compositions of MTCs (YIE *et al.*, 2012), as well as in automatic recommendations of transformations (ROCCO *et al.*, 2015). The information shown in Figure 4.36, line 8, is only descriptive. However, we acknowledge from practice that even in the same format of XMI, UML tools usually do not export models in the same format, implying in interoperability issues, which are only detected too late at runtime. Therefore, even if not used for computation

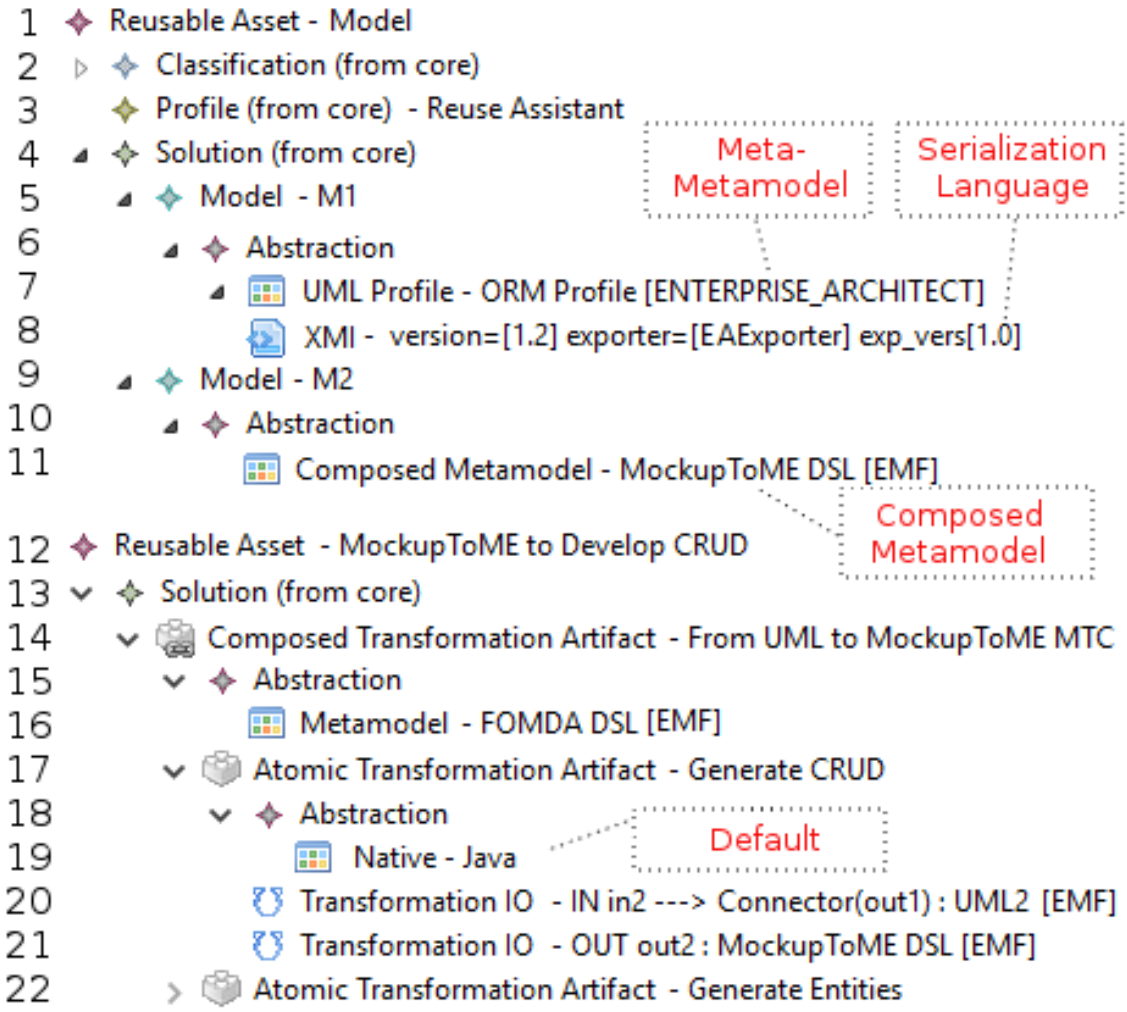


Figure 4.36: Representing abstractions for models

purposes, semantics associated with models and transformations are relevant for decision making.

Finally, the standard RAS allows to represent other semantics associated with artifacts (e.g., examples, tutorials, documentation, etc). Although RAS allows to represent nested artifacts, as those exemplified between lines 14 and 22 for components and sub-components, we consider that it is important to correctly structure these elements conforming to MDE Settings. Likewise, the OCL rules embedded in our tool support for asset design help software engineers to represent well structured settings, avoiding wrong constructors for artifacts. Thus, this demonstration allows to conclude that RAS++ is better structured and representative for a pivotal language for MDE as a Service than RAS.

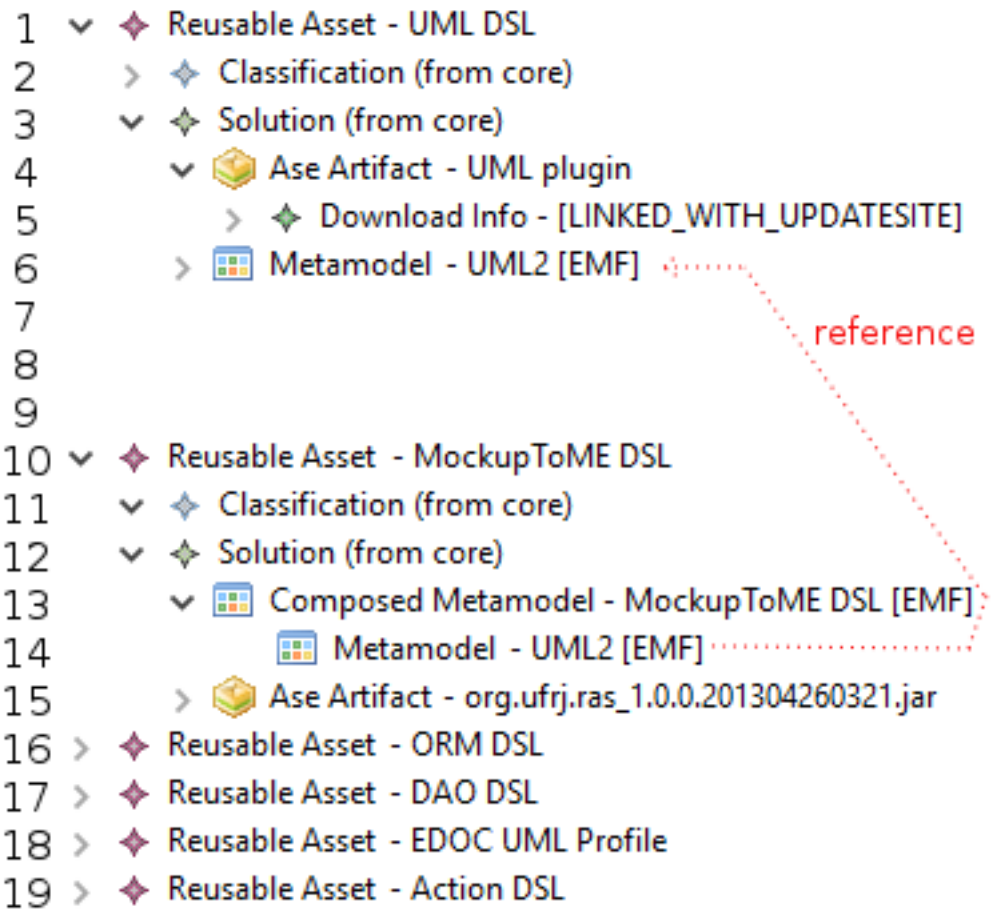


Figure 4.37: Assets representing DSLs

DSLs and Meta-Metamodels

This section expands the running example for the representation of assets considering a complete solution for system engineering of information systems. Likewise, the exemplified MDE Setting is a piece of the lifecycle of MockupToME Method, which is more complex and allows to represent all the concepts introduced in RAS++ DSL. This method includes the following resources that should be represented in assets: 10 UML Profiles, 12 DSLs generated with the Eclipse EMF (3 are represented as instances of `Metamodel` and 9 as `ComposedMetamodel`), 343 MDE Artifacts, 152 libraries that support the execution of model-based tasks and 11 documents that are used by teams such as tutorials. For more information, a technical definition presents the configured method (BASSO *et al.*, 2016d) and adopted for the development of a real web information system.

Figure 4.37 shows some assets that represent data associated with DSLs used in MockupToME Method. From line 1 to 6 information associated with the asset "UML DSL" is presented, which represents data from the following artifacts: "UML Plugin", which encapsulates information from jars from UML2/MDT accessed through an

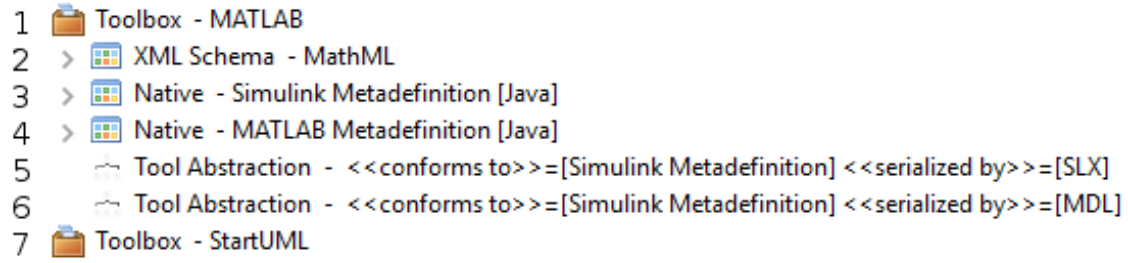


Figure 4.38: Representing some design toolboxes with RAS++.

update-site; and the "Metamodel UML2" that encapsulates information from the "uml2.ecore" file. The asset shown between lines 10 and 15 shows information from the MockupToME DSL. Line 14 shows a reference from the composed metamodel for the UML 2 metamodel, as required to configure the scenario of GUI DSLs from the MockupToME Method. Other assets from lines 16 to 19 classify other DSLs used in this method, thus complementing our case study for the representation of assets.

Toolboxes

Figure 4.38 shows some toolboxes represented conforms to RAS++. MATLAB for example, is composed by many design tools. Since artifacts can be composed, an a toolbox is an artifact in RAS++, one can insert any artifact in toolboxes. For example, lines 16-18 shows three meta definitions found in this toolbox. Lines 19 and 20 exemplify two channel adapter represented as serialization formats. These channels are available in MATLAB to persist models designed with the Simulink tool: SLX and MDL. Design tools typically provide channel adapters as serializations for input and output. This type of channel particular of MDE Toolboxes are only possible for representation in RAS++, therefore a contribution in comparison to AMS and RAS.

It is also important to mention that information about channel adapters is considered useful for integration purposes in many research areas such as: 1) Model transformation chain (YIE *et al.*, 2012) and model tool chain (LIEBEL *et al.*, 2014); 2) service oriented tool integration (BIEHL *et al.*, 2014); 3) Messaging channel APIs (FRANTZ and CORCHUELO, 2012) and; 4) Application Lifecycle Management (ALM) tools (ZAKHEIM, 2017).

Moreover, design tools have been shared in repositories including: MDEForge (ROCCO *et al.*, 2015), ReMoDD (FRANCE *et al.*, 2007) and SHARE (GORP and MAZANEK, 2011). In this sense, the representation of toolboxes in assets becomes essential for a common representation language built on asset concepts.

4.3.4 Final Remarks

We have represented a total of 43 assets that classify artifacts associated with the MockupToME Method (BASSO *et al.*, 2016d). Five assets are for classification of models generated with the MockupToME Method. These assets are represented similarly to those shown in Figure 4.36, line 1 to 11. For DSLs, 22 assets are represented, each one considering one metamodel with keywords for search and instructions for use. They are represented similarly to those illustrated in Figure 4.37. Other 16 assets were represented considering artifacts and settings for model transformation components, as those illustrated in Figure 4.35. Finally, two instances of repositories are represented, dividing these assets in a local and global knowledge base.

These assets classify artifacts, resulting in a total of: 152 Java libraries with resource locators for EMBEDDED_AS_FILE and LINKED_WITH_MAVEN, 343 instances of `AtomicTransformationComponent` and `MTRule`, 11 documents/tutorials, 22 metamodels (.ecore files) plus 22 plugins (a pack of .jar files as LINKED_WITH_UPDATESITE). The core information from these artifacts was automatically generated with the help of our tool prototype, which will be presented elsewhere. Thus, our effort was to correct the generated specifications, divide artifacts in appropriate assets and provide descriptive semantics that are not possible to be generated.

Our conclusion is that RAS++ is representative to operate as a pivotal language for MDE Artifacts and Settings for internal issues. Complemented by our tool support, which helps software engineers on the representation of assets, we also conclude about the feasibility in terms of representativeness for implementation of coopetition in MDE as a Service. Therefore, based on the generality of RAS++ and the specificity introduced in support for MDE Artifacts and Settings, we demonstrated that the introduced concepts have applicability and contribution for the motivated scenario.

The next chapter provide evaluations considering issues in coopetition scenarios.

Chapter 5

Assessments

> Truth is what stands the test of experience.
>
> Albert Einstein

Previous chapter considered the representation of reusable assets from our research group. In order to evaluate whether RAS++ fits to the motivated scenario, which is bound by artifacts shared in global repositories such as SEMAT (JACOBSON *et al.*, 2012), GEMOC (COMBEMALE *et al.*, 2014) and ReMoDD (FRANCE *et al.*, 2007), this chapter presents four complementary assessments: Section 5.1 depicts the first assessment, considering 81 asset representations by mining data from a real global repository for MDE; Sections 5.2, 5.3 and 5.4 details three analytical studies for the motivated scenario.

5.1 Mining ReMoDD Repository

This section considers a case study on mining reusable assets from the ReMoDD (FRANCE *et al.*, 2007) repository <www.cs.colostate.edu/remodd/v1/>, registered between 2011 and 2016, and answers the following research question: **Q6: Is RAS++ representative to play the role of a common language for the assets available in the ReMoDD repository**?

We selected ReMoDD repository and not GEMOC (COMBEMALE *et al.*, 2014) because the last one stores artifacts in ReMoDD. SEMAT (JACOBSON *et al.*, 2012) is not considered for evaluation because it contains only artifacts for process engineering, which is out of the scope of the current version of RAS++. MDEForge (ROCCO *et al.*, 2015) is not considered because the web platform was inoperative until the end of 2016. SHARE (GORP and MAZANEK, 2011) is not considered because it was found recently in our mappings. Besides, ReMoDD is a repository of MDE assets that has been widely disseminated by experts associated

with some important conferences, such as MODELS, SPLC, ASE, GPCE, among others. Currently, it shares a total of 81 assets described in Table 5.1 and Table 5.2.

This repository has emerged as reference for sharing information about design tools, including download links and reference web-pages, as well as physical files for models derived from case study, metamodels, transformations and others. Due to its goal of promoting a global reuse scenario, ReMoDD contains a diversity of MDE artifacts that can be integrated into a software project developed with the support of MDE resources (i.e., in an arbitrary tool chain). In other words, its artifacts can be used to evaluate the representations suggested in the phases *Specification*, *Acquisition* and *Transformation* of RAS++. ReMoDD is, therefore, the best candidate for evaluation of the RAS++ representativeness considering hybrid representations.

Four types of assessment studies are described by Empirical Software Engineering: Case Study, Survey, Experiment and Action Research. Our evaluations considered only two types: Case Study and Action Research, the last one adopted for just two of our studies. RUNESON and HÖST (2008) states that case studies scope well organized studies in the field to small toy examples (proof-of-concepts), so different taxonomies should be adopted for characterization. The term case study is also used to describe a field study and observational study.

To RUNESON and HÖST (2008), such studies can be characterized by using one or more types of research methodologies:

> *"**Exploratory**-finding out what is happening, seeking new insights and generating ideas and hypotheses for new research; **Descriptive**-portraying a situation or phenomenon; **Explanatory**-seeking an explanation of a situation or a problem, mostly but not necessary in the form of a causal relationship; **Improving**-trying to improve a certain aspect of the studied phenomenon."* (RUNESON and HÖST, 2008)

We divided this case study in five different evaluations, focusing in a specific preliminary phase for tool chain and using different research protocols:

1. Section 5.1.1 presents an evaluation of assets for the *Specification* phase trough an **Exploratory study**, which is used in the next evaluations.

2. Section 5.1.2 presents a **Descriptive study** of explicit data in ReMoDD for the *Transformation* phase, without our intervention in representations.

3. Section 5.1.3 highlights our findings concerning implicit data associated with technicalities for the *Transformation* phase in a **Descriptive study**, performed without our interventions.

Table 5.1: Assets shared in ReMoDD between 2011 and 2016 (Part I)

Asset	Asset Name	Year
A01	Automated Provisioning of Customized Cloud Service Stacks	2015
A02	OpenCompare case study	2015
A03	UML Models Generated and Derived for Evaluating aToucan	2014
A04	Renarrating Metalanguage Integration	2014
A05	Toward a Megamodeling Approaches Overview	2014
A06	Integrating modeling and programming languages - The case of Java and fUML (case study models)	2014
A07	uml-profile-store	2014
A08	OpenStack Model	2014
A09	Class diagram of OpenNebula	2014
A10	How Could Ancient Romans Know About UML Statecharts?	2014
A11	Modeling the Architecture and Design of the Crisis Management System Product Line Using SimPL	2013
A12	Modeling Specification for bCMS Product Line using Feature Model, Component Family Model and UML	2013
A13	Modeling Crisis Management System with the Restricted Use Case Modeling Approach	2013
A14	Modeling Car Crash Management with KAOS	2013
A15	RELAX/SysML/KAOS	2013
A16	CMA@RE SysMLKaosVersionRemodd.pdf	2013
A17	Coloured Petri Net Model of the bCMS system using CPN Tools	2013
A18	Behavior as-is and to-be and Goal-Belief models	2013
A19	bCMS Case Study: FAMILIAR	2013
A20	Requirements Modeling in SEAM: The Example of a Car Crash Management System	2013
A21	Using AMoDE-RT and DERAF to specify a Crisis Management System – Complete Model Description	2013
A22	Complete Goal-Belief	2013
A23	Witness Goal-Belief	2013
A24	Behavior to-be	2013
A25	Behavior as-is	2013
A26	URML Model of bCMS (HTML Export)	2013
A27	MontiArcAutomaton BumperBot Models	2013
A28	Webmail MTS model	2013
A29	Webmail Example Specification	2013
A30	Multi-models to aid Decision Making in Enterprises	2013
A31	SensApp DSML composition case study: From Sensors to Visualization Dashboards	2013
A32	Railroad Crossing Management System	2013
A33	Applying BPMN on bCMS	2013
A34	UML-UseCaseDiagram-AssessmentForm(post-workshop)	2012
A35	A Catalog of UML Model Transformations	2012
A36	Updated Activity Theory bCMS Model Description for CMA-2012	2012
A37	Umple submission for Comparing Modeling Artifacts workship at Models 2012	2012
A38	Concern-Driven Development with AoURN and RAM	2012
A39	Models for bCMS using AspectSM	2012
A40	Intentional Requirements Engineering	2012
A41	The VCL Model of the Barbados Crisis Management System	2012
A42	bCMS in LEAP	2012
A43	Modeling with Adapt Cases	2012
A44	CMA12 - CMS Domain- i* - Group 6	2012
A45	Aspect-Oriented Modeling for Performance Evaluation with UML+MARTE, LQN, and CSM	2012
A46	UML-RT to kiltera model transformation and examples	2012
A47	SAM Metamodel in Ecore format + OCL well-formedness rules	2012
A48	HRC (Heterogeneous Rich Components) Metamodel in Ecore format + OCL Well-formedness rules	2012
A49	RBAC metamodel in Ecore format + OCL Well-formedness rules	2012

Table 5.2: Assets shared in ReMoDD between 2011 and 2016 (Part II)

Asset	Name	Year
A50	ER 2 RE Metamodel in Ecore format + OCL well-formedness rules	2012
A51	Declarative Workflow Metamodel in Ecore format + OCL Well-formedness rules	2012
A52	CPFSTool Metamodel in Ecore format + OCL well-formedness rules	2012
A53	SAD3 Metamodel + OCL Well-formedness rules	2012
A54	B Language Metamodel in Ecore format + OCL well-formedness rules	2012
A55	OMG Common Warehouse Metamodel in Ecore format + OCL well-formedness rules	2012
A56	OMG Diagram Definition metamodels + OCL well-formedness rules	2012
A57	CORBA Component Model Specification Metamodels in Ecore format + OCL well-formedness rules	2012
A58	OCL specification metamodels in Ecore format + well-formedness rules specified in OCL	2012
A59	MOF 2.0 Metamodels in Ecore format + OCL well-formedness rules	2012
A60	UML 2.2 Packages in Ecore format + OCL well-formedness rules	2012
A61	Comparison Criteria for bCMS Models of CMA Workshop	2012
A62	bCMS - Requirements Definition	2012
A63	Activity Theory Comparison Criteria - Dec 2011 version of criteria	2011
A64	Reusable Aspect Models for the bCMS Case Study	2011
A65	Activity Theory Models for the bCMS Case Study - CMA@MODELS2011	2011
A66	bCMS case study models for OO-SPL approach	2011
A67	bCMS-SPL case study: A proposition based on the Cloud Component Approach.	2011
A68	Model Driven Service Engineering applied to bCMS	2011
A69	bCMS Case Study: AoURN	2011
A70	DT4BP to TimedCaaFWrk model transformation	2011
A71	UML Class Diagram Patterns	2011
A72	AoURN and RAM models of the Crisis Management System	2011
A73	DT4BP - Meta-Model	2011
A74	DT4BP - Syntactic Definition	2011
A75	TimedCaaFWrk - Coordinated Atomic Actions Meta-Model	2011
A76	Elevator Control System	2011
A77	Video Conferencing System	2011
A78	Models of the ODP specification of the PhoneMob system	2011
A79	MOBIES Powertrain Models	2011
A80	NewCarCrashSPL	2011
A81	CMS:Lassy:REACT SPL Feature Model and Feature Mapping	2011

4. Section 5.1.4 evaluates representations for transformation components by mining hidden data from research papers associated with assets.

5. Finally, Section 5.1.5 groups previous studies to reach out new analysis and conclusions with an **Explanatory study**.

5.1.1 Evaluation 1 - Representation of Explicit Contextual Data

According to Whittle et al., "Finding the right problem is crucial" for applying MDE in the industry (WHITTLE *et al.*, 2015). The authors claim that there is little information regarding best practices and tools recommended for each end-user context. A first look in ReMoDD repository shows that descriptive information is packed in hybrid assets, stored in a database populated by researchers and practitioners of MDE (FRANCE *et al.*, 2007). An hybrid asset represents data associated with

MDE Artifacts of different natures (e.g., built on different languages and tools and with different intents) and are shared by different sources (individual, organization, research groups). For example, an asset can be composed of models, process tasks, APIs and libraries, binaries and tools, model transformations, etc (FRANCE *et al.*, 2006). Therefore, this class of assets are supposed to be represented in ReMoDD.

In this thesis, we expanded the application of reusable assets concepts, usually applied to software artifacts (e.g., models, components, source code, etc.) (WERNER *et al.*, 2009), to hybrid assets that can be represented with RAS++ in at least three preliminary phases for tool chain. The *Specification* phase introduced the RAS++ metamodel with extensions for the RAS metamodel, unifying representational concepts also from AMS. In this phase, MDE specialists specify many reusable assets for classification with data for federation, thus focusing in details for classification, search and retrieval.

Goal

Our main goal in this study is to evaluate the quality of the information from ReMoDD in terms of hybrid assets focusing in the *Specification* phase. We wanted to evaluate whether RAS++ is representative for ReMoDD assets. In affirmative case, this would suggest the relevance of metaclasses available for the execution of *Specification* phase, thus providing index for validity.

Table 5.3: Representation of explicit contextual information

Analyze	The amount of descriptive data introduced in Assets for MDE Artifacts
With the purpose of	Characterizing
Regarding	Hybrid classifications for MDE
From the viewpoint of	Researcher and domain expert
In the context of	A global reuse repository for MDE Assets, i.e., ReMoDD.

The specific goal is shown in Table 5.3. The first asset representation problem we address concerns the classifications adopted by ReMoDD. We want to answer the following questions:

- *Q1 - What are the classification groups associated with assets in ReMoDD?* Our objective is to map the classifications adopted by the repository.
- *Q2 - Is RAS++ able to represent classifications from ReMoDD?* Our objective is to evaluate the representativeness of RAS++ for MDE assets.
- *Q3 - Which assets from ReMoDD are associated with an industrial context and with an academic context?* The objective is to categorize industrial and academic endeavors. Since ReMoDD is not a platform for asset negotiation and,

because companies adopting MDE usually consider their resources extremely sensitive and confidential (a business differential), we suspect that industry is not actively participating in this initiative.

- *Q4 - What constitutes MDE Artifacts referenced in assets from ReMoDD?* MDE Resources and software development contexts/scenarios used in ReMoDD may not be representative of those occurring in other realistic MDE settings. Different software development processes, domains and organizations may lead to different needs from MDE Resources. Thus, our objective is to reach what is available for asset customers.

- *Q5 - Concerning intents for use in a software development project, what constitutes data from assets in ReMoDD?* The goal is to map some assets for future process engineering purposes. Due to the focus of this thesis, we intentionally removed from RAS++ elements added in support for representations found in a mapping of contributions for process engineering. However, it is our long term goal to reintroduce these representations, thus needing a mapping.

- *Q6 - Concerning technicalities from MDE Artifacts, what constitutes data from assets in ReMoDD?* Our objective is to understand whether some technical information is provided for selection of artifacts as well as on the quality of this data.

Research Method

In order to classify assets, we adopted metrics for descriptive statistics analysis using qualitative and quantitative data. Our population, or focus group, is composed by 81 assets, thus representing the amount of elements shared by ReMoDD. Moreover, RUNESON and HÖST (2008) conducted a descriptive survey from Mining Software Repositories (MSR) (D'AMBROS *et al.*, 2008), used as basis for the execution of this study. So, we adapted his empirical procedure on mining of the ReMoDD repository, following the evaluation plan composed by:

1. The first step to mine data from repositories is to define a *Data modeling*, which in the MSR protocol is defined as a common representation for software artifacts. In our case, this data modeling is an asset representation. For the data modeling we used the recent version of the generated DSL through the RAS++ metamodel.

2. The second step is called *Data retrieval and processing*. We used our generated Eclipse Plugin for representation of all the 81 Assets extracted from ReMoDD database. We focused exclusively on the manual representation of descriptive

data discussed in the *Specification* phase. We, therefore, applied the criteria for classification (as recommended by RAS and AMS), resulting in representations as the one shown in Figure 5.2.

3. The last step is *Data analysis*, which is characterized by the analysis of modeled data. Thus, we have extracted quantitative and qualitative attributes based on explicit data associated with assets to identify how hybrid they are.

Analysis

We present our analysis according to the research questions of the evaluation as follows.

Q1 - What are the classification groups associated with assets in ReMoDD?

Figure 5.1 presents six possible classifications of representations in ReMoDD. Classifications are divided in groups and detailed in items, as presented in Table 5.4.

Table 5.4: Qualitative analysis of descriptor groups found in ReMoDD

Descriptor Group	Description	Usage	Value
Artifact Development Contexts	The asset provider points out the usage/production contexts	Mandatory	Prefixed
Artifact Types	The asset provider itemizes the types/nature of artifacts, describing the asset as encapsulating model, metamodel, and other artifact types	Mandatory	Prefixed
Required Tool	The asset provider informs in items which tools are needed for use/adoption of shared artifacts	Optional	Free Form
Programming /Modelling Languages	The asset provider represents which programming languages or modelling languages are used for purposes of production/processing of the shared artifacts	Optional	Free Form
Modelling Languages /Notations	The asset provider describes modeling languages/DSLs/notations used to produce the artifacts, i.e., considering that these artifacts are models	Optional	Free Form
Lifecycle Phases	The asset provider represents in which phases of a lifecycle the artifacts are relevant	Optional	Free Form
Software /System Domains	The asset provider itemizes software or system domains from the shared artifacts	Optional	Free Form
Keywords	A general group used for keyword search purposes	Optional	Free Form

More than one descriptive item can be used in any of eight classification groups. Except for groups *Artifact Development Contexts* and *Artifact Types*, the other ones are optional. This means that asset providers can share MDE Artifacts without any extra description. We grouped *Artifact Types* in with the following prefixed possibilities: Metamodel/Profile, Model, Model Transformation, Tool, Methodology/Technique/Process, Case study, Lecture/Tutorial/Training, Requirements Document, Pattern Catalog and Assessment. *Artifact Development Contexts* are prefixed in: Industry, Academia, Research Project, Workshop/Focus Group and Challenge Problem/Competition.

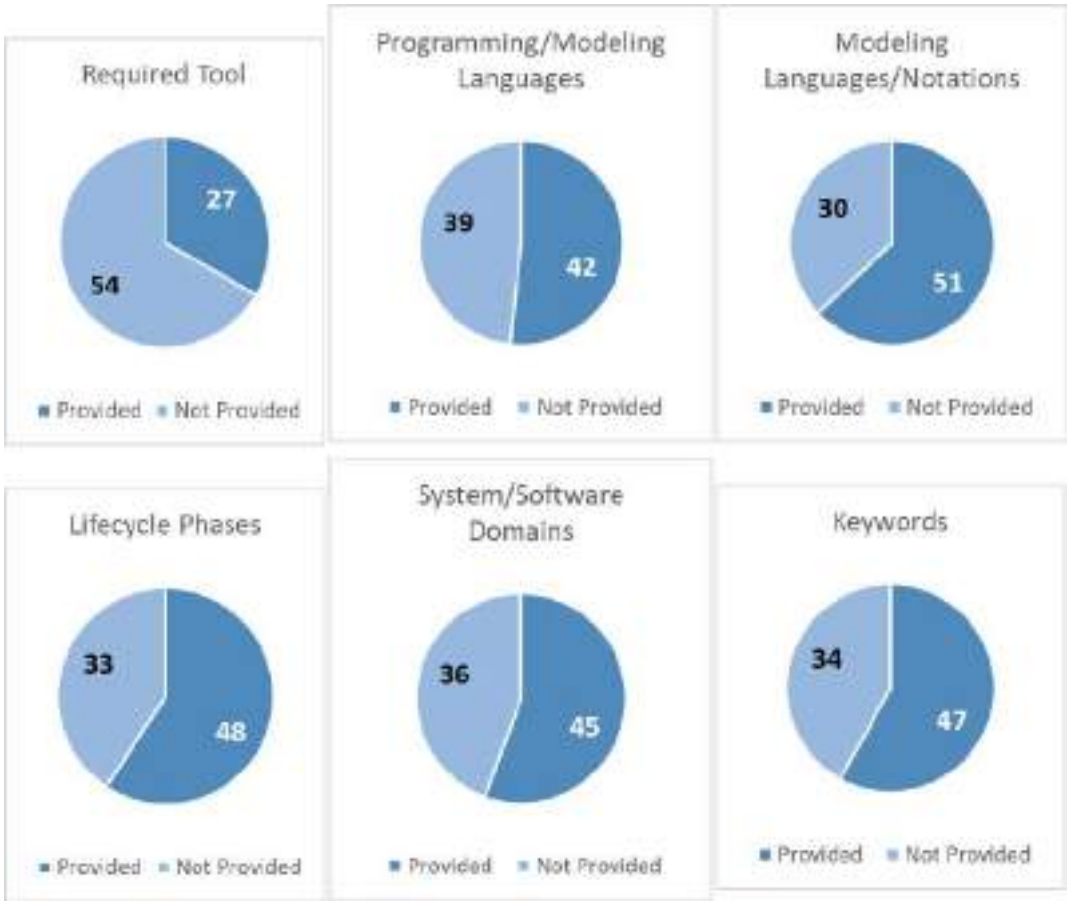


Figure 5.1: ReMoDD - Quantitative analysis of classification groups.

From these 81 assets, we found that: 1) Only 27 assets represent information on tools that are required to use/adopt the shared artifacts; 2) Information about programming/modelling languages is found only in 42 assets and about modelling languages/notations in 51. An issue found here is ambiguity. Since it is not possible to understand the difference between these two groups, we found that much of the descriptive information is repeated in both classifications from these assets; 3) A total of 48 assets map information about in which lifecycle phases they are relevant and; 4) More than half, i.e., 45 assets, point out exactly for which software/system domains they are relevant.

Q2 - Is RAS++ able to represent classifications from ReMoDD?

Through these groups we represented 81 assets using five RAS++ metaclasses, see Table 5.5. Figure 5.2 shows the classification for the first asset **{A01}**: "Automated Provisioning of Customized Cloud Service Stacks". In order to characterize a group of classifications, and in conformity with the RAS++ metamodel, we represented instances of `DescriptorGroup` and instances of `FreeFormValue` to itemize the descriptive data in each group. It is important to mention that such metaclasses were extracted from RAS, not requiring extensions.

Each asset is inserted into an instance of `Repository` or `ServiceProvider`.

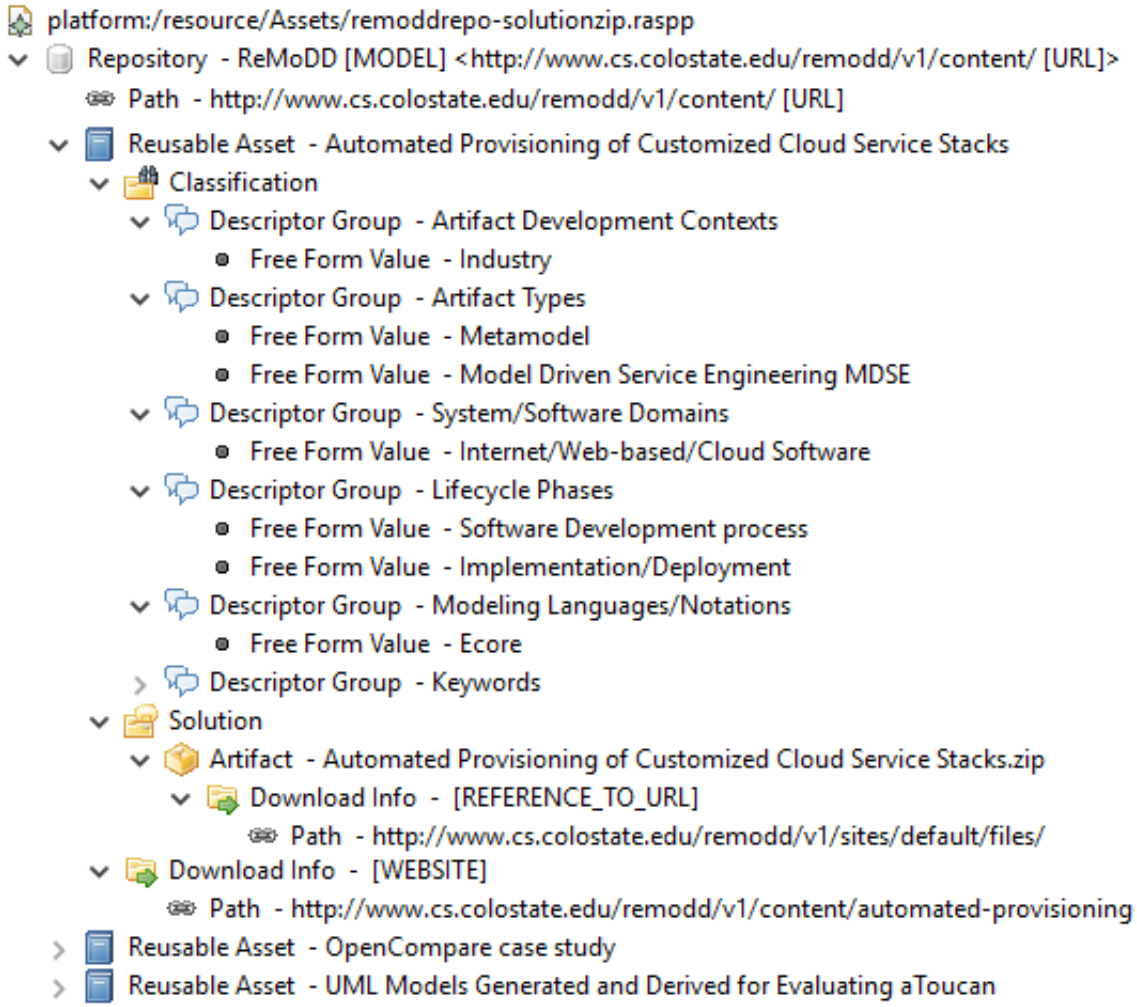


Figure 5.2: Screenshot of ReMoDD assets designed conforms to RAS++

Assets and artifacts contains independent URLs for resource locators represented as instance of `Resource`, a metaclass extracted from AMS. In AMS, artifacts and assets are instances of Resource. A simple asset selected to exemplify the representation of resource locators is shown in the bottom-part of Figure 5.2. A resource locator is an instance of *DownloadInfo*, describing the protocol for downloading if any, and `Path`, representing a link. The asset **{A47}** is more complex since it is composed by the following resource locator data (sum of instances of DownloadInfo and Path): one pair for the asset itself (instance of `ReusableAsset`); 28 pairs for representation of 28 inner artifacts (instances of `Artifact`).

It is possible to have more instances of `Path` than `DownloadInfo`, but not the opposite. For example, assets **A13**, **A30** and **A40**, present more instances of `Path` than of *DownloadInfo*. In this case, besides locators for artifacts and asset, they provided simple links (instances of Path) for tool support.

The representation of assets allowed us to answer affirmatively to the question *Q2*. In other words, in the phase *Specification*, RAS++ matches the needs from

Table 5.5: Counting of representations for each asset by RAS++ Metaclass

LEGEND [Asset = Asset, DescriptorGroup = DG, FreeFormValue = FFV, Artifact=ART, DownloadInfo = DI, Path = Path]											
Asset	**DG**	**FFV**	**ART**	**DI**	**Path**	**Asset**	**DG**	**FFV**	**ART**	**DI**	**Path**
A01	6	18	1	2	2	A02	6	9	1	2	2
A03	8	13	1	2	2	A04	8	14	1	2	2
A05	8	18	1	2	2	A06	8	19	2	3	3
A07	8	15	1	2	2	A08	7	7	1	2	2
A09	6	7	1	2	2	A10	4	9	1	2	2
A11	8	15	1	2	2	A12	8	21	1	2	2
A13	7	13	1	2	3	A14	6	8	1	2	2
A15	8	15	2	3	4	A16	5	6	1	2	2
A17	8	14	2	3	3	A18	7	17	1	2	2
A19	8	13	2	3	3	A20	7	16	1	2	2
A21	7	16	2	3	3	A22	7	8	1	2	2
A23	6	8	1	2	2	A24	7	9	1	2	2
A25	6	9	1	2	2	A26	8	13	3	4	4
A27	7	11	3	4	4	A28	5	5	1	2	2
A29	5	6	1	2	2	A30	7	11	1	2	5
A31	3	3	1	2	2	A32	5	7	2	3	3
A33	4	8	1	2	2	A34	3	3	1	2	2
A35	6	6	1	2	2	A36	5	8	1	2	2
A37	5	7	7	8	8	A38	7	14	1	2	2
A39	5	5	1	2	2	A40	8	9	2	3	5
A41	8	11	1	2	2	A42	7	9	1	2	2
A43	4	7	1	2	2	A44	7	9	1	2	2
A45	6	17	1	2	2	A46	8	27	2	3	3
A47	2	2	28	29	29	A48	2	2	5	6	6
A49	2	2	5	6	6	A50	2	2	3	4	4
A51	2	2	4	5	5	A52	2	2	3	4	4
A53	2	2	3	4	4	A54	2	2	2	3	3
A55	2	2	22	23	23	A56	2	2	6	7	7
A57	2	2	4	5	5	A58	2	2	4	5	5
A59	2	2	6	7	7	A60	5	6	1	2	2
A61	3	9	1	2	2	A62	5	11	1	2	2
A63	5	8	1	2	2	A64	6	7	2	3	3
A65	5	11	2	3	3	A66	8	11	2	3	3
A67	7	14	1	2	2	A68	6	13	1	2	2
A69	7	13	1	2	2	A70	6	8	1	2	2
A71	5	15	1	2	2	A72	3	4	19	20	20
A73	3	4	1	2	2	A74	2	2	1	2	2
A75	3	3	1	2	2	A76	2	2	4	5	5
A77	2	2	8	9	9	A78	3	4	1	2	2
A79	4	6	39	40	40	A80	4	4	2	3	3
A81	2	3	1	2	2	-	-	-	-	-	-

ReMoDD repository.

Q3 - Which assets from ReMoDD are associated with an industrial context and with an academic context?

Figure 5.3 shows our analysis of 111 descriptive data focused on the group "Artifact Development Context" from 81 assets. Some assets associate more than one artifact development context. We can highlight that from 81 assets, only 4 of them (5%) are from an industrial endeavors **{A01, A13, A74 and A79}** and 77 assets

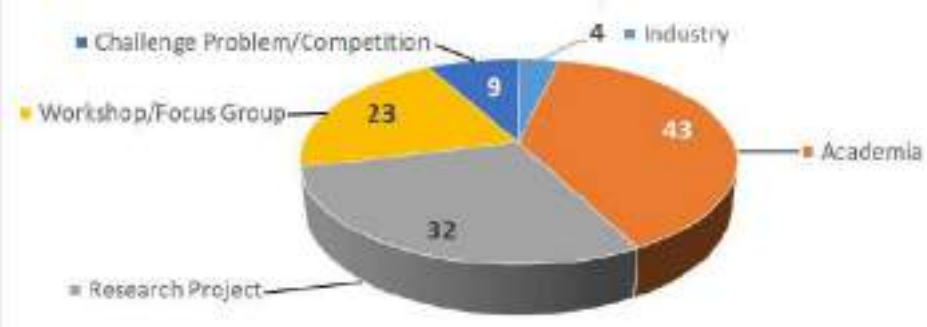


Figure 5.3: ReMoDD - Quantitative analysis of artifact development context.

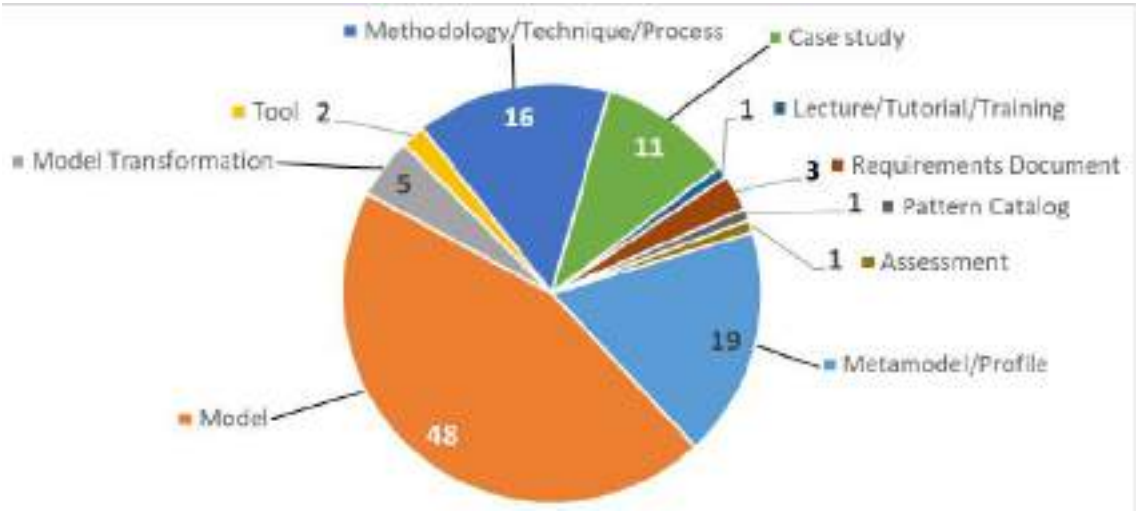


Figure 5.4: ReMoDD - Quantitative analysis of artifact types.

are from academic endeavors (95%). Therefore, all the effort to promote global reuse through ReMoDD has no meaningful inclusion from the industry.

It is important to mention that the inclusion of industry was a primary goal of the ReMoDD proposal: *"We aim to collect and make available MDD artifacts from industry, academia, and other public domain sources (e.g., artifacts produced by open-source projects)"* (FRANCE *et al.*, 2006). This suggests that, although the research has been motivating the inclusion from industry, available data in ReMoDD presents a hard truth that challenges future business implementation in MDE as a Service: industrial participation is not a reality in state of the practice. Therefore, in order to align new initiatives for MDE that also target industrial participation such as GEMOC (COMBEMALE *et al.*, 2014), MDEForge (ROCCO *et al.*, 2015) and SEMAT (JACOBSON *et al.*, 2012), an effort to understand the lack of cooperation from industry is important.

Q4 - What constitute MDE Artifacts referenced in assets from ReMoDD?

In order to answer Q4, we considered information about the nature of artifacts (their types, constructors and associated tools) shared in this repository. Figure 5.4

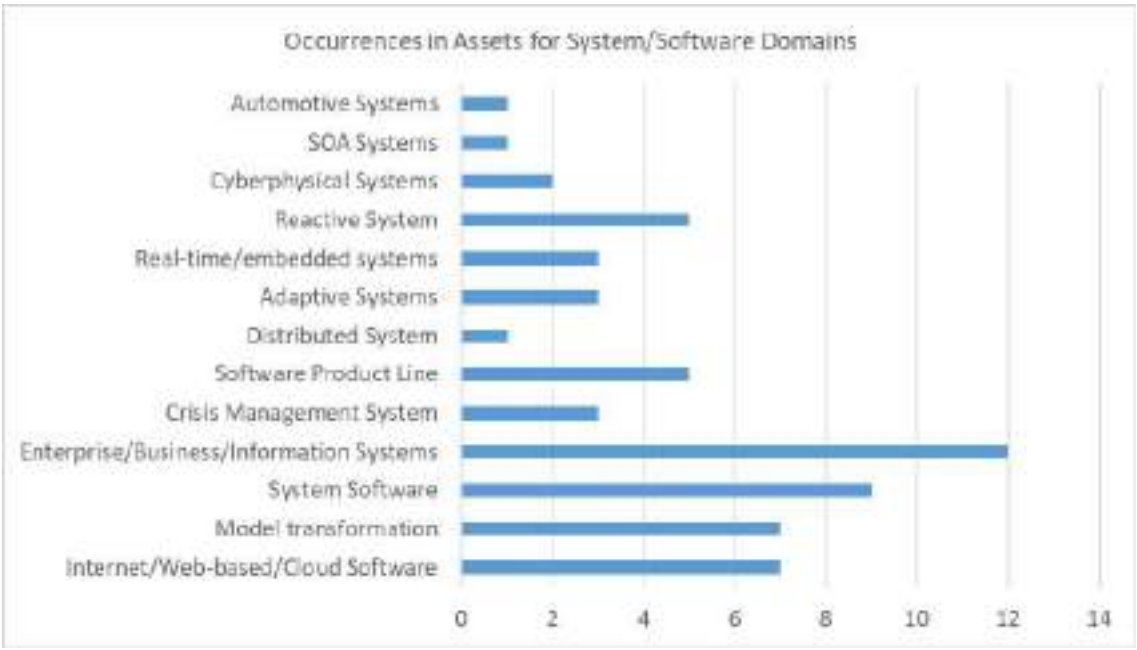


Figure 5.5: ReMoDD - Quantitative analysis of system/software domains.

shows a graph quantifying 107 descriptive items for the group "Artifacts Types". Through this classification, we found that many MDE Artifacts reflect what has been used in realistic MDE Settings found in the literature. Asset consumers may include these artifacts (i.e., Model, Metamodel/Profile, Model Transformation and Tool) in an integration approach. For example, a model transformation chain or an automated software development process may include artifacts referenced by most assets (69%): 74 occurrences. Besides, 32 occurrences (31%) are descriptive data suggesting didactic material for execution of some tool chain approach (i.e., Methodology/Technique/Process, Case study, Lecture/Tutorial/Training, Requirements Document and Pattern Catalog). We conclude that artifacts configure realistic data, built from different sources and natures, thus providing opportunities for reuse and integration in tool chains.

Q5 - Concerning intents for use in a software development project, what constitutes data from assets in ReMoDD?

In order to answer this research question, we provided detailed analysis for two descriptor groups that characterize hybrid assets: System/Software Domains and Lifecycle Phases. Aligning data from the software domain and the lifecyle phase, it is possible to infer whether some artifacts are relevant for inclusion in a MDE-based software project. Therefore, concerning *Q5*, the following data suggest the use of hybrid assets in ReMoDD:

- 48 assets divided in eight different lifecycles for software development, as shown in Figure 5.6;

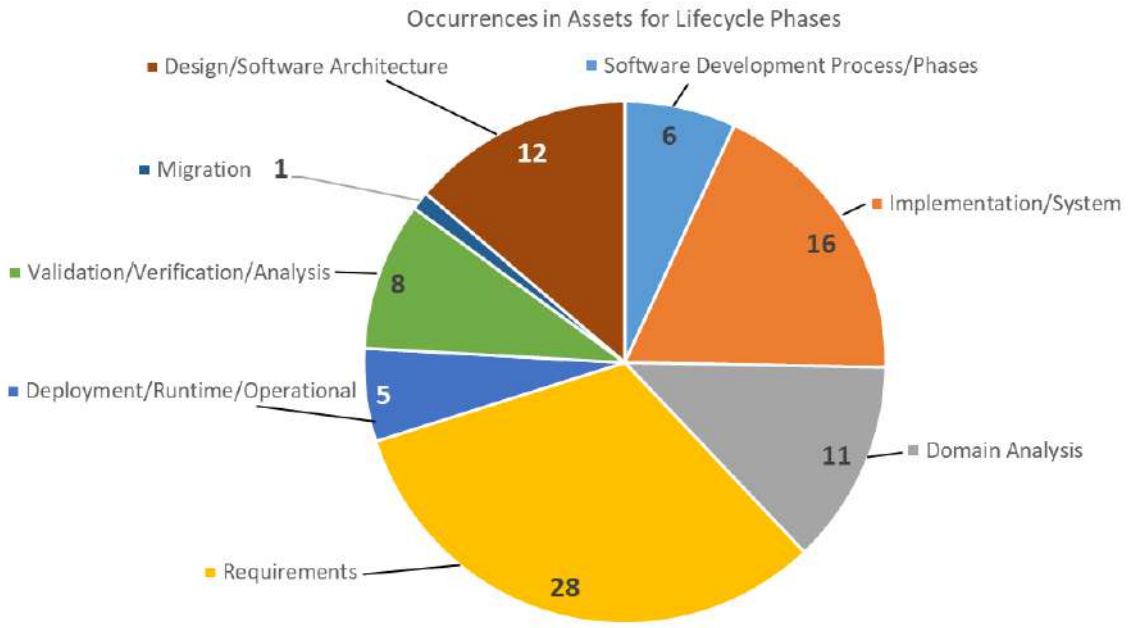


Figure 5.6: ReMoDD - Quantitative analysis of lifecycle phases.

- 45 assets divided in 13 system/software domains, as shown in Figure 5.5. Two classifications are used in more than one software engineering discipline (*Software Product Lines* and *Crisis Management System*) and the others usually relate to the discipline *Analysis and Design*.

Figure 5.6 shows 87 occurrences for software development lifecycle phase found in 48 assets. These occurrences highlight which lifecyle phase the content of an asset is relevant and they seem to be extracted from a software development process model, such as OpenUP <http://epf.eclipse.org/wikis/openup/>. In this sense, assets associated with the group *Software Development Process/Phases* provide support for process engineering, while the other groups suggest the relevance of assets for a specific phase. This information suggests that assets have diverse applications in many phases from a software development context, thus attempting to help the execution of more than one software engineering discipline. This finding is important in the motivated MDE as a Service scenario because some assets can cooperate, i.e., can be used together to assist a MDE-based software process, while others can be competitors in a specific phase.

We found that asset providers have little criteria when classifying MDE Artifacts. For example, classifications of Software/Application Domain and Lifecycle Phase are very useful for decision making in a selection process. Their usage on assets could be interesting for a tool chain specialist to decide whether some asset is relevant or not for a specific need in a software project. However, we found ambiguities in the provided data.

Q6 - Concerning technicalities from MDE Artifacts, what constitutes

data from assets in ReMoDD?

In this section we focused in an analysis of data that could be used by a technical stakeholder searching for information related with model, metamodel, transformation, design/refinement/transformation tool. This question is relevant to understand whether some classification is provided for supporting the selection of artifacts of different natures.

Figure 5.5 shows 59 occurrences from 45 assets concerning the group "System/Software Domains". We found that this group is represented with ambiguous data. For this reason, we filtered and divided them between 13 well-defined classifications. This information is relevant for a preliminary phase of tool chain in MDE. For example, concerning a software development context from Adapit, as discussed in Section 3.1, it possible to mention that 12 assets from 81 are relevant (right-on-the-target) for consideration in an eventual integration: those associated with the value "Enterprise/Business/Information Systems", including **{A07, A18, A20, A22, A23, A24, A25, A26, A40, A42, A78 and A80}**. However, the data is still very abstract to understand whether shared artifacts are really relevant for a specific need in a software development context, thus needing in-depth analysis after a list is returned by the search engine.

Conclusions

We found ReMoDD's classification groups well tailored for MDE Artifacts, but two are ambiguous: *Programming/Modelling Language*s and *Modelling Languages/Notations*. However, it is possible to understand the purpose and application of shared artifacts. This is one of the goals of this repository and, from our point-of-view, it is accomplished. Most information that should be explicit in repository structures is available as physical files in documents, papers and tutorials, which requires long time for asset consumers to find and analyze adequate options for MDE Artifacts.Therefore, some recent classifications are proposed for describing types of model transformations (CRIADO *et al.*, 2015; CZARNECKI and HELSEN, 2003; DA SILVA, 2015; LÚCIO *et al.*, 2014; WHITTLE *et al.*, 2015) and could complement the demonstrated group descriptors as well.

So far, ReMoDD failed to attract data from industry. We found that there is an expressive number of assets shared by academic endeavors (77), but an inexpressive number shared by industrial endeavors (4). FRANCE *et al.* (2007) originally proposed ReMoDD with the goal to connect these two endeavors, where opportunities for reuse could be played by collaborators and competitors in model-driven development (FRANCE *et al.*, 2006). As suggest the data shared between 2011 and 2015, this goal is not reached along four years effort. This is quite surprising since one third of ReMoDD collaborators are from industry: Microsoft Research, USA;

Siemens Corporate Research, Worldwide; Deere & Company, USA; Tecknowledge, USA; Eaton Innovation Center, Worldwide; IBM, Canada; Motorola, Worldwide. Thus, an open question is why has ReMoDD failed to attract industry interest?

Recent research has been motivating the implementation of coopetition, also through repositories, as a possible solution to attract industrial interest. For example, some in software ecosystems (SECOs) perspectives (BADAMPUDI *et al.*, 2016; DOS SANTOS *et al.*, 2013) and others in collaboration for MDE (ROCCO *et al.*, 2015, 2016) have been claiming that reuse repositories such as ReMoDD are not an adequate platform for collaboration, cooperation and competition. These contributions have been considered by us to elaborate representations from the "Acquisition" phase. Thus, they may help to answer this open question in the next evaluations.

Representations for assets adopted by ReMoDD are less representative than those found in RAS (HONG-MIN *et al.*, 2009) and component reuse repositories (DOS SANTOS and WERNER, 2010): 1) Artifacts are mere links for downloading physical files, while in RAS they can be annotated with semantics for context and other descriptive data independently from the asset data; 2) While in RAS the metaclass `RelatedAsset` allows the representation of important data that characterizes whether some asset is complementary of competitive for an arbitrary tool chain, ReMoDD provides no way to represent relationships between assets; 3) There is no activity that supports the reuse of artifacts in ReMoDD, while in RAS these activities are well structured for usage and adaptation before the introduction in a target context (e.g., a tool chain). Such representations are highlighted as benefit for component reuse in some studies (DOS SANTOS and WERNER, 2010; ELIAS *et al.*, 2006; OMG, 2005). However, they could not be evaluated in this study. This means that RAS++, through the metaclasses `ArtifactActivity`, `Activity`, `VariabilityPoint`,`VariabilityPointBinding`, and others, is more representative than ReMoDD, thus matching the representational needs.

Although the original proposal for ReMoDD is to represent assets for model-driven development, we found that assets are not restricted for systems engineering. This way, ReMoDD can be classified as a Knowledge Base for MDE in a more general term (SELIC, 2014): as resources for Model-Based Development (MBD).

The knowledge built in this repository lacks modernization for recent advances found in literature. For example, recent contributions built on AMS/OSLC (ELAASAR and NEAL, 2013; ROCCO *et al.*, 2015) allow establishing asset relationships through clouds and local repositories, which is called federation (BADAMPUDI *et al.*, 2016). Due to this limitation, ReMoDD artifacts are stored locally, without the support for resource locators across repositories. This limitation from ReMoDD hampered our evaluation on the diversity of representations for artifact federation, which means that RAS++, through the metaclass

`DownloadInfo`, is more representative than ReMoDD.

We considered the group of classifications proposed in ReMoDD very useful for searching assets mapped for some MDE Artifact specificity, such as values adopted in group *Artifact Development Context.* However, we found that there is important hidden knowledge inside artifacts shared in ReMoDD, or available in some associated publication, that should be represented in assets. This way, we have identified three open questions related to ReMoDD that should be investigated by this thesis:

- Are these classifications adequate for execution of the *Acquisition* phase? We believe that classifications are too superficial in comparison with data recommended for decision making (BADAMPUDI *et al.*, 2016).
- Are these classifications and artifacts adequate for the representation of technicalities found in the *Transformation* phase? Classifications are general and have no purpose as pivotal representation, thus opening a window for detailing technicalities associated with MDE Artifacts and Settings in RAS++.
- In terms of integration/tool chain, are the shared artifacts adequate for execution of *Transformation* phase? We have found possible tool chain approaches to be used in integration, thus candidate for data transformations.

This study allowed us to conclude about the representativeness of RAS++ from explicit data available in the state of the practice for MDE: ReMoDD repository. Abstractions included in RAS++ for the *Specification* phase are suitable for the data provided by this repository, matching all the representational needs. In this study we found a considerable diversity in languages used to build models, metamodels and transformations, developed in support for some software development contexts. ReMoDD is, therefore, rich in hybrid reusable assets.

5.1.2 Evaluation 2 - Representation of Explicit Technical Data

In this study we aim to mine the technical information associated with the 81 assets of the RemoDD repository. In the following, we describe the protocol of this study.

Goal

Our main goal in this study is to evaluate the quality of information in terms of technical aspects for the execution of the *Transformation* phase. Assets should be found in repositories such as ReMoDD without the concern of integration itself. In any case, we suspect that the asset providers represent components used in a tool

chain approach with a kind of technical information allowed by ReMoDD. In the affirmative case of availability of these data, this would suggest the relevance of the artifacts available for integration, providing index of validity and selection of focus groups in this repository.

Table 5.6: Representation of explicit technical information

Analyze	The amount of descriptive data introduced in Assets for MDE Artifacts
With the purpose of	Characterizing
Regarding	Technicalities for MDE
From the viewpoint of	Researcher and domain expert
In the context of	A global reuse repository for MDE Assets, i.e., ReMoDD.

- *Q1 - Which technicalities are found in assets?*
- *Q2 - Which assets are candidate for a tool chain approach?*

Research Method

Our population, or focus group, is composed by 81 assets, thus representing the amount of elements shared by ReMoDD. Thus, we applied the same protocol as in the previous study, but now focusing on the extraction of technical data.

Analysis

This sections reports our analysis of **Q1 - Which technicalities are found in assets?**

Figure 5.7 shows 66 occurrences within 41 assets representing semantics for the "Programming/Modeling Languages" group. These occurrences are divided into 38 languages. UML is the most used for the representation of artifacts (13 for UML plus 4 for UML2). However, this data changes when considering the other ambiguous group "Modeling Languages/Notations".

With regard to 41 assets of the group "Programming/Modeling Languages", Figure 5.8 shows four references to the classification of "Toolboxes/Frameworks " and 15 occurrences for classification "Model Transformation Languages". These are descriptive data that give semantics to the MDE resource types, which we also consider as good candidates for integration into an MDE-based software development project/process.

Figure 5.9 presents data on diversity in notations and modeling languages, showing options that can configure a hybrid scenario in MDE. We found 48 assets divided into: 27 assets semantically connected with only one "programming/modeling language", not covering the definition of a tool chain (BATORY *et al.*, 2013b); 21

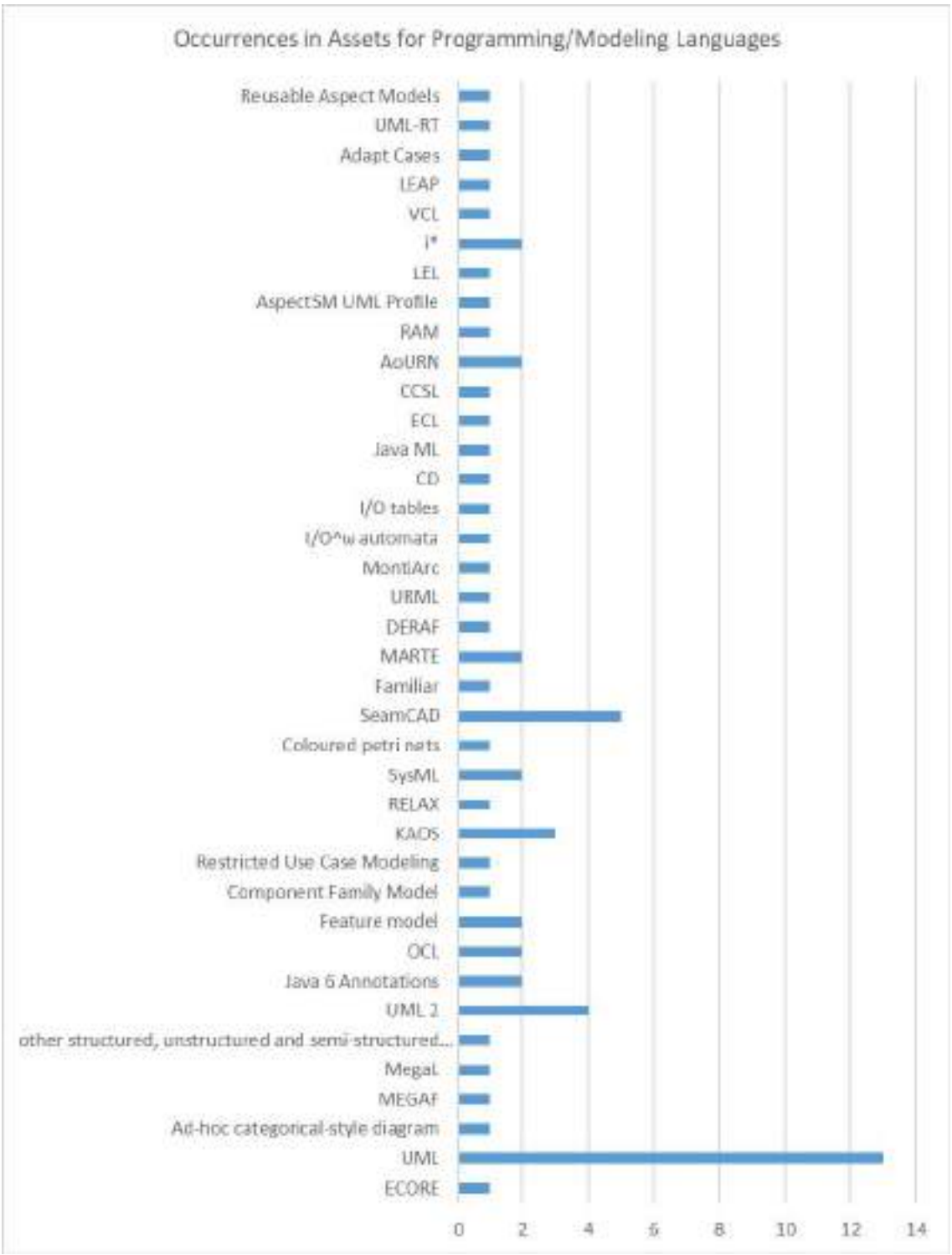


Figure 5.7: ReMoDD - Quantitative analysis of DSLs.

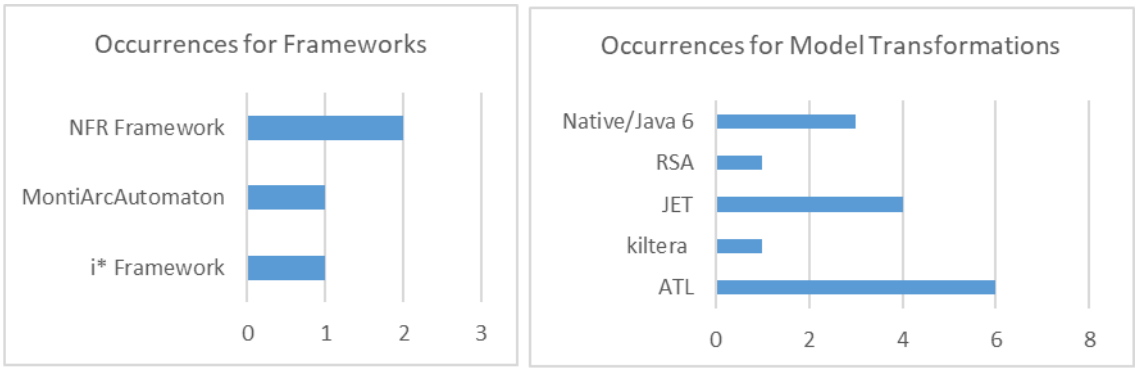


Figure 5.8: ReMoDD - Quantitative analysis of technicalities for tool chain.

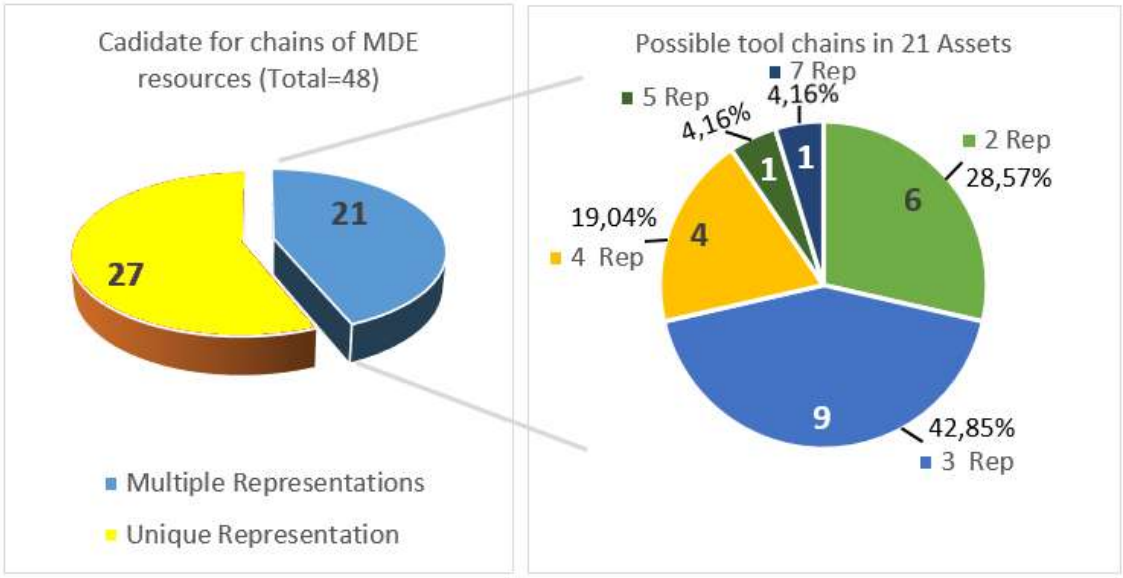


Figure 5.9: ReMoDD - Quantitative analysis of candidate assets for tool chain.

resources suggesting the use of two or more Tools/DSLs, therefore candidates for tool chains.

Although it is not an objective of this study to represent in-depth information on tool chain relationships, the explicit data of these 21 assets have semantics suggesting a possible integration: six tool chains composed of two languages, as suggested by the descriptive data of assets **{A06, A07, A17, A29, A43, A79}**; nine tool chains composed of three languages, as suggested by the data collected from assets **{A05, A12, A15, A16, A30, A32, A39, A40 and A60}**; four tool chains composed of four languages, as suggested by assets **{A10, A21, A38, A46}**; a tool chain of asset **{A45}** using five languages and; a tool chain of asset **{A27}** using seven languages.

Of the 27 assets analyzed using only one programming/modeling language, we find 22 that explicitly refer to the use of UML: **{A03, A05, A06, A07, A08, A09, A10, A11, A12, A21, A29, A35, A38, A39, A43, A45, A46, A60,**

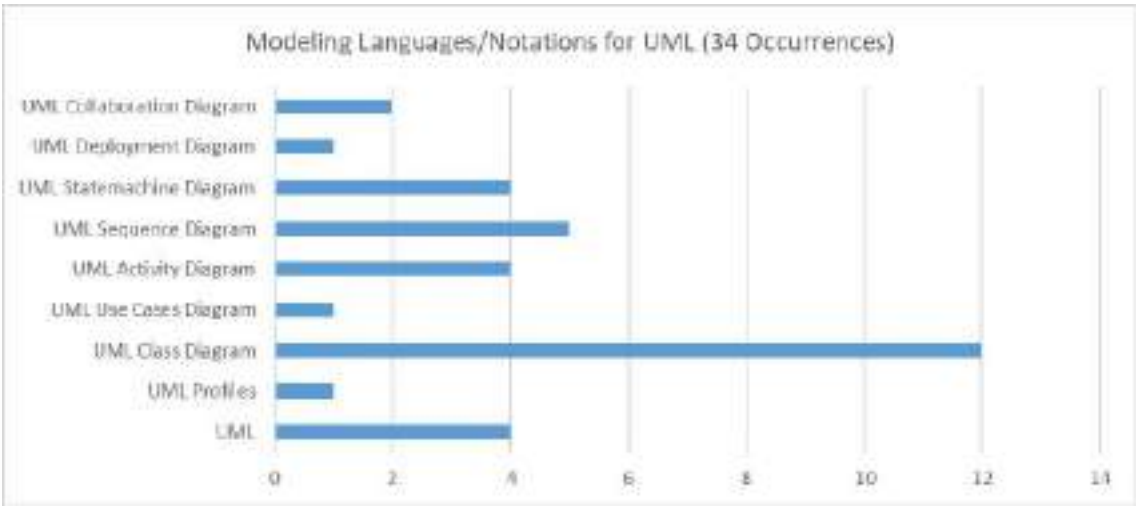


Figure 5.10: ReMoDD - Quantitative analysis of UML.

A66, A68, A71 and A72}. These assets are analyzed in Figure 5.10 considering which diagram/notation is used. Five assets provide only reference to the use of UML/Profile, not describing which notation is used, while 17 explain the adopted notation through taxonomy. We can highlight the use of the class diagram notation, with 12 occurrences of 17 possible. These data are in line with the results of industrial research on the UML adoption (AGNER *et al.*, 2013; HUTCHINSON *et al.*, 2011; PETRE, 2013): the class diagram is most commonly used among UML notations.

Figure 5.11 shows the number of occurrences for semantics of DSLs, that is, semantics that do not include UML usage. In this sense, it is possible to notice that six assets refer Ecore as a modeling language for classes instead of UML, which means that Ecore has been used to represent architectural models of systems (not metamodels). This is an interesting finding because in (PETRE, 2013) it is reported that most companies use some modeling tool to represent the class diagram only. Similarly, as shown in Figure 5.10, the UML class diagram is most commonly used notation.

Q2 - Which assets are candidate for a tool chain approach?

ReMoDD repository is relevant to research and practice in MDE. However, in our previous survey we lacked an analysis of explicit information that could be used in technical decision making, involving tool chain contexts. So, to answer *Q2 - Which assets are candidate for a tool chain approach?*, we have refined our initial analysis to find out the reuse opportunities.

To this end, we review previously extracted data in a new analysis for data triangulation (RUNESON and HÖST, 2008), clustering data from descriptor groups "Required Tool", "Programming/Modeling Languages" and "Modeling Languages/Notations". We then quantify the occurrences for domain-specific languages

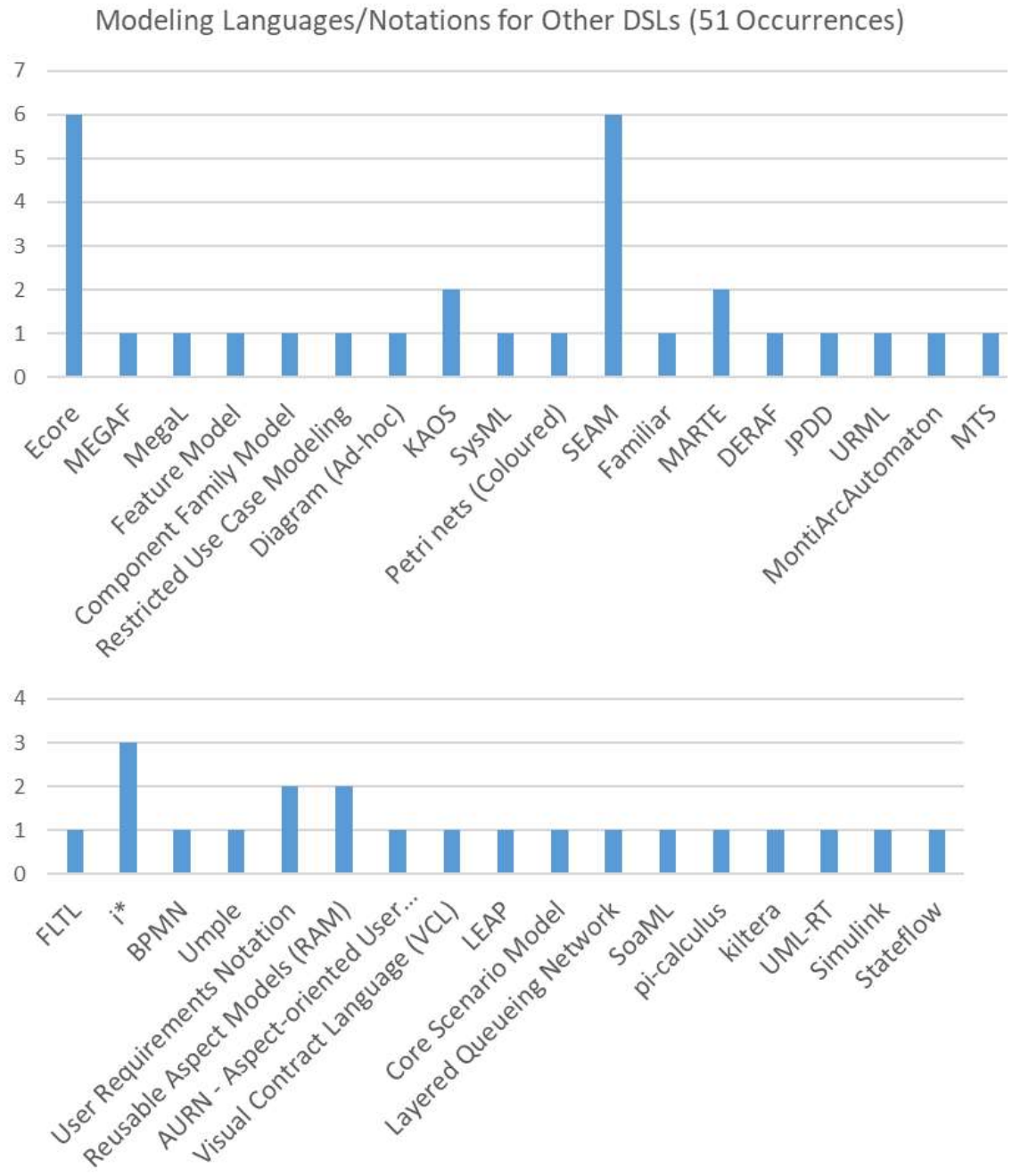


Figure 5.11: ReMoDD - Quantitative analysis of DSLs.

and/or tools by exporting models in some open format between (XML, JSON, Text) and/or programming languages used for the development of model transformations. In the end, we draw the following conclusions.

Figure 5.9 shows 48 assets as candidates to incorporate some kind of tool chain. From these assets, it is noticed that most use a homogeneous set of DSLs and tools for model representation (27 items), while the semantics of 21 of them suggests to consider a heterogeneous scenario. Of the 27 assets that consider a single representation language, 22 items have semantics associating them with UML. Even though strictly considering the use of UML, these 22 assets have hybrid notations composed of the following semantics:

- Eight assets highlight the use of more than one UML notation: five assets use two UML notations **{A06, A10, A38, A46 and A68}**; one asset is using three UML notations **{A66}**, and; two assets are using four UML notations **{A03 and A45}**;

- Eight assets jointly use UML and other DSLs, as shown in Figure 5.11: **A05**, including `MegaF` and `MegaL`; **A12**, including `Feature Model` and `Component Family Model`; **A21**, including `MARTE`, `DERAF` and `JPDD`; **A29**, including `FLTL`; **A38**, including `User Requirements Notation`, `Reusable Aspect Models (RAM)`, and `Aspect-oriented User Requirements Notation`; **A39**, including `AspectSM` and `MARTE`; **A45**, including `Core Scenario Model`, `Layered Queueing Network`, `MARTE` and `SoaML`, and; **A46**, including `pi-calculus`, `kiltera` and `UML-RT`.

We also analyze the use of UML in the context of `System/Software Domains`, represented only in 14 assets, as follows: four covering the Model Transformation domain **{A03, A05, A35, A45 and A46}**; two covering the Crisis Management System domain **{A11, A12}**; for covering the Software Product Line (SPL) domain **{A12 and A66}**; four covering the Internet/Web-based/Cloud Software domain **{A07, A08, A09 and A29}**; **{A06}** covering the System Software domain; two covering the Reactive System domain **{A38 and A46}**; **{A45}** covering the Service Oriented Architecture (SOA) systems **{A45}**, and; **{A46}** covering two other domains (Embedded Software and Real-time embedded systems).

Final Remarks

We extracted 81 artifacts available for use in REMoDD between 2011 and 2016, representing them as resources and extracting quantitative data on resources associated with tool support. We found that data associated with assets do not support structural features of tool chains. There are some links to tool support, but not

a structured representation used in integration. The technical aspects for decision making are represented, but at a semantic level that can not be processed by transformations that aim to promote automatic integration.

It was not found assets for the domain of information systems that reports the use of UML. This finding is consistent with PETRE (2013), suggesting that UML has some resistance to this context. For example, Figure 5.5 shows that at least 12 items have artifacts designed with DSLs not built into UML Profiles.

Every asset is represented independently by sharing MDE Artifacts of different types and purposes, as shown in Figures 5.4 and 5.6. Models are built using different languages, as shown in Figures 5.7, 5.10 and 5.11. Figure 5.8 shows that at least five languages are used in the development of model transformations and some tools / frameworks are required for their execution. Therefore, ReMoDD contains hybrid assets.

Small descriptions relevant to technical decision-making in chain of tools are also found. As Figure 5.11 illustrates, design tools are created using different meta-generators, such as EMF (Ecore notations), or simply developed manually, such as some in Java **{A08, A12, A17, A19, A33}**, or C++ **{A08}**, or other programming language. This relationship (for example, models and transformations constructed in metamodels) is not explicit, requiring knowledge and hard work to discover these important details for the integration. Based on our previous experiences in integrating design tools developed without the support of a meta-generator, models are typically imported and exported in different formats (XMI, JSON, text files). Due to the need to develop transformations of models classified by LÚCIO *et al.* (2014) as *Serialization Intention*, this adds a lot of extra effort to integrators.

5.1.3 Evaluation 3 - Mining MDE Artifact Hidden Data

This case study is focused on mining MDE Artifacts of the 81 assets, thus needing to discover the expert's technical knowledge. Through this mining, we hope to discover technical aspects associated with these artifacts, such as models, metamodels and transformations.

Goal

As our previous studies have pointed out that the technical structural features of MDE Artifacts are not detailed in ReMoDD, our objective here is to represent this hidden/non-explicit data. We want to evaluate whether the hidden data found in MDE Artifacts can be represented with RAS++. If so, this would suggest the relevance of the available metaclasses for the execution of the *Transformation* phase, providing a validity index.

Table 5.7: Mining non-explicit data from MDE Artifacts

Analyze	The amount of technical data associated with MDE Artifacts
With the purpose of	Characterizing
Regarding	Hybrid classifications for MDE
From the viewpoint of	Researcher and domain expert
In the context of	A global reuse repository for MDE Assets, i.e., ReMoDD.

The specific goal is shown in Table 5.7. The third asset representation problem we address concerns the representation of technical aspects using structural features proposed in RAS++ for MDE Artifacts. We want to represent hidden data in ReMoDD to answer the following questions:

- *Q1 - What are the MDE artifacts referenced in ReMoDD repository assets?* Our goal is to identify which artifacts the research community is sharing in ReMoDD.
- *Q2 - Is RAS++ able to represent the hidden data of MDE artifacts from the ReMoDD repository?* Our goal is to evaluate the representativeness of RAS ++ for MDE Artifacts.
- *Q3 - How many occurrences for technical metaclasses (e.g., those inheriting MDEArtifact) are found?* This quantification of metaclasses would allow us to understand if RAS++ is relevant for the representation of technicalities of MDE artifacts.

Research Method

In order to answer aforementioned research questions, we have adopted metrics for descriptive statistics analysis using qualitative and quantitative data. Our population, or focus group, is composed by 81 assets, thus representing the amount of elements shared by ReMoDD. Moreover, RUNESON and HÖST (2008) have conducted a descriptive survey from Mining Software Repositories (MSR) (D'AMBROS *et al.*, 2008), used as basis for the execution of this study. So, we adapted their empirical procedure on mining the ReMoDD repository, following the evaluation plan composed by:

1. We used the recent version of the generated DSL for representation of MDE Artifacts, thus starting the first step called *Data modeling* based on a previous representation for assets.
2. For the step *Data retrieval and processing* we passed by all the 81 Assets previously represented. We found artifacts represented as Zip and leaf extensions. We mined the content hidden inside Zip files, such as the one shown in

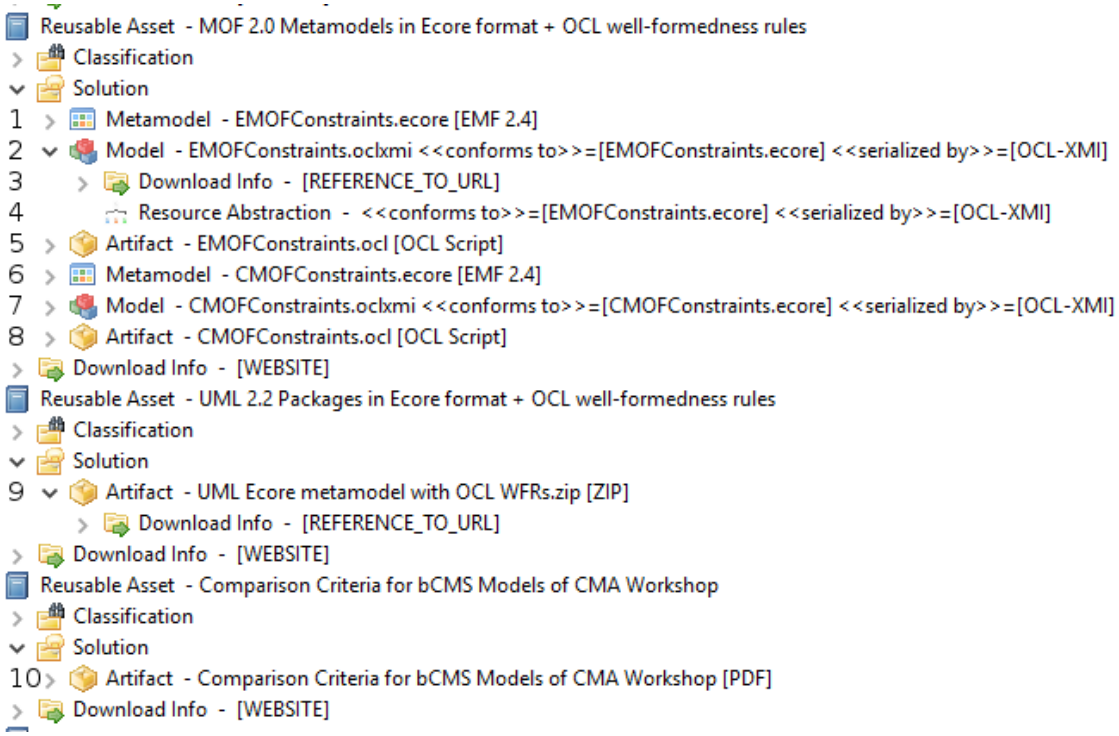


Figure 5.12: MDE Artifacts from three of ReMoDD' assets.

Figure 5.12 line 9. We considered for analysis only data non associated with examples or descriptions. In other words, we ignored the processing of artifacts for documentation or demonstration (txt, pdf, doc, gif, jpeg, spreadsheet), as the one shown in Figure 5.12 line 10.

3. The *Data analysis* step is characterized by the analysis of modeled data. Only metadata associated with MDE Artifacts are considered, as those shown in Figure 5.12 between lines 1 and 8. Through this data type, we have extracted quantitative attributes.

Analysis

Q1 - What are the MDE artifacts referenced in ReMoDD repository assets?

We found 20 assets with leaf extensions that indicated their nature as MDE Artifacts: **{A37, A47, A48, A49, A50, A51, A52, A53, A54, A55, A56, A57, A58, A59, A70, A73, A75, A76, A77 and A79}**. It is important to mention that ReMoDD represents such artifacts as any other for documentation, thus not properly representing the specificity of MDE. Because the artifact metadata cannot be automatically mined, this data type is considered hidden. We found 15 assets with hidden data inside ZIP files: **{A01, A02, A03, A06, A07, A08, A11, A17,**

A19, A27, A30, A34, A46, A60 and A62} . Therefore, our analysis include 35 assets.

Q2 - Is RAS++ able to represent the hidden data of MDE artifacts from the ReMoDD repository?

Table 5.8 shows instances of artifacts represented for each of 35 analyzed assets. Not so surprisingly, we found a unique instance of `TransformationComponent` and another from `Toolbox` in two different assets `A46` and `A70`. The low interest in sharing for free this type of artifact suggests that asset providers may consider these artifacts as some business differential. This remind us from a recent position from Jan Bosch at ICSE 2017: there is no good will in free assets shared by industry. From this point of view we conclude that, if there is no gain for asset providers in sharing assets, then their interest in ReMoDD reduces as well.

Table 5.8: Technical representations for each asset by RAS++ Metaclass

LEGEND [Asset = Asset, Metafacility = MM, Model = MO, Toolbox = TB, Transformation Component = TC, MDE Artifact=MA]

Asset	MM	MO	TB	TC	MA	Asset	MM	MO	TB	TC	MA
A01	4	0	0	0	0	A02	5	5	0	0	0
A03	0	16	0	0	0	A06	0	6	0	0	0
A07	19	0	0	0	0	A08	0	1	0	0	0
A11	0	1	0	0	0	A17	0	1	0	0	0
A19	0	9	0	0	0	A27	0	6	0	0	0
A30	0	3	0	0	0	A34	0	1	0	0	0
A37	0	6	0	0	0	A46	0	0	1	0	0
A47	1	1	0	0	26	A48	1	1	0	0	2
A49	1	1	0	0	2	A50	1	1	0	0	1
A51	2	1	0	0	1	A52	1	1	0	0	1
A53	1	1	0	0	1	A54	1	1	0	0	0
A55	1	1	0	0	20	A56	2	2	0	0	2
A57	2	2	0	0	0	A58	2	2	0	0	0
A59	2	2	0	0	2	A60	13	0	0	0	13
A62	0	3	0	0	6	A70	0	0	0	1	0
A73	1	0	0	0	0	A75	1	0	0	0	0
A76	0	1	0	0	0	A77	0	4	0	0	0
A79	0	18	0	0	0	-	-	-	-	-	-

Artifacts as instances of `Metadefinition`, `Model` and `MDEArtifact`, the last one used to represent OCL and other scripts, are abundant. Instances of `Metadefinition` are constructors for DSLs, totalizing 61 artifacts distributed in 19 assets: A01, A02, A07, A47-A60, A73, A75. We found 107 artifacts as instances of `Model`, which demonstrates that, indeed, ReMoDD has been used for academic purposes.

Instances of `Model` must be associated with instances of `SerializationLanguage`, as illustrates Figure 5.12, line 4. We did not find models serialized in JSON, JMI and RDF, but we found serialization in Binary, Plain text, XML and XMI. This way, Figure 5.13 shows our analysis of 23 different exporters associated with 107 models.

XMI	XML	RDF	JSON	PLAIN_TEXT	JMI	OTHER (Binary)
8	9	0	0	5	0	1

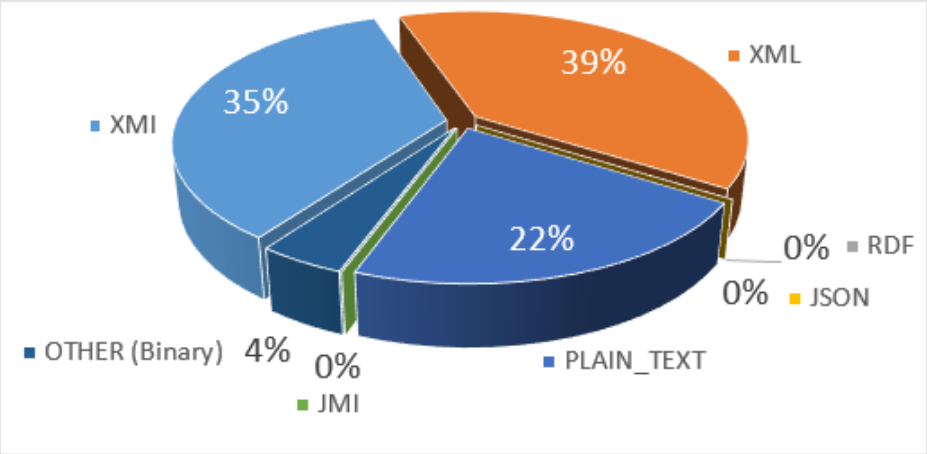


Figure 5.13: Serializations used by MDE Artifacts.

As Figure 5.14 demonstrates, RAS++ is able to represent the technical data about Toolboxes that ReMoDD does not represent. These are structural data used by integration tools and can be used in the **Transformation** phase, using automatic integration mechanisms. As suggests ZAKHEIM (2017), hidden technical knowledge is a threat to integration in a tool chain environment. Differently from ReMoDD, RAS++ exposes the hidden knowledge about structural feature from MDE Artifacts including metamodels, toolboxes and dependencies. This, therefore, is a benefit found in RAS++ but not in the ReMoDD repository, which means that for a feasible coopetition implementation (i.e., a less costly approach for integration), ReMoDD needs to be reconstructed.

The last research question from this third mining study is: **Q3 - How many occurrences for technical metaclasses (e.g., those inheriting MDEArtifact) are found?** We answer this question by analyzing data about the number of representations as an instance of `AtomicTransformationArtifact`, `MDEArtifact`, `MetaDefinition`, `MetaMetadefinition`, and `Toolbox`. The results are shown in Figures 5.15, 5.16 and 5.17.

Figure 5.15 highlights internal representations, whose content we find directly (looking inside the contents of the shared artifacts), and external representations, which are semantic links to tools that we had to look for on the web. For example, Figure 5.18 shows classifications represented in the previously mined assets. For artifacts within an asset containing one of these semantics, we went beyond the repository to describe the associated tool, thus representing one or more Toolbox instances. This effort demonstrates how difficult it is for an integrator to perform a reuse of shared artifacts in global repositories. Since the information required

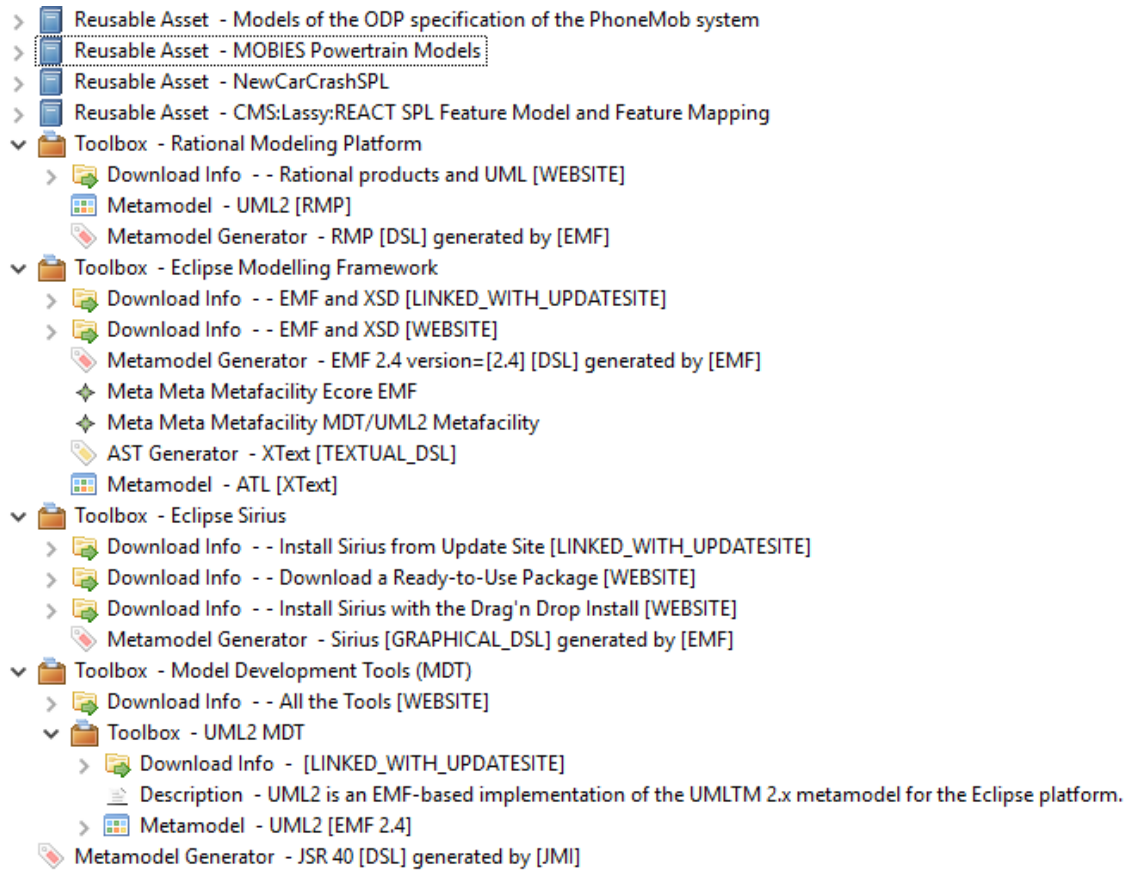


Figure 5.14: Data expressed externally to assets used by MDE Artifacts.

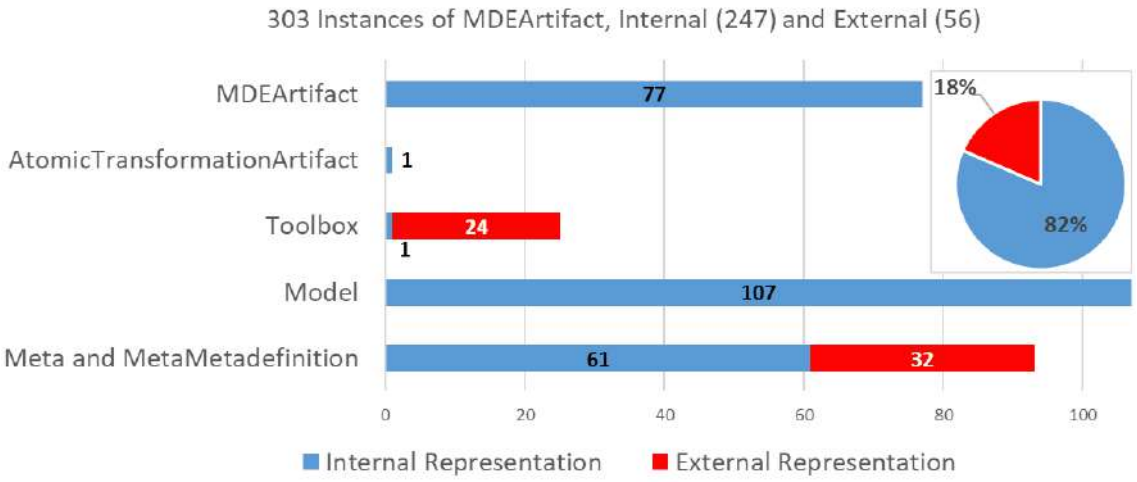


Figure 5.15: All the technical details extracted from the hidden data.

AdHocDSL	Metamodel	ComposedMetamodel	Native	XmlSchema	Grammar	UMLProfile
3	47	0	3	5	3	19

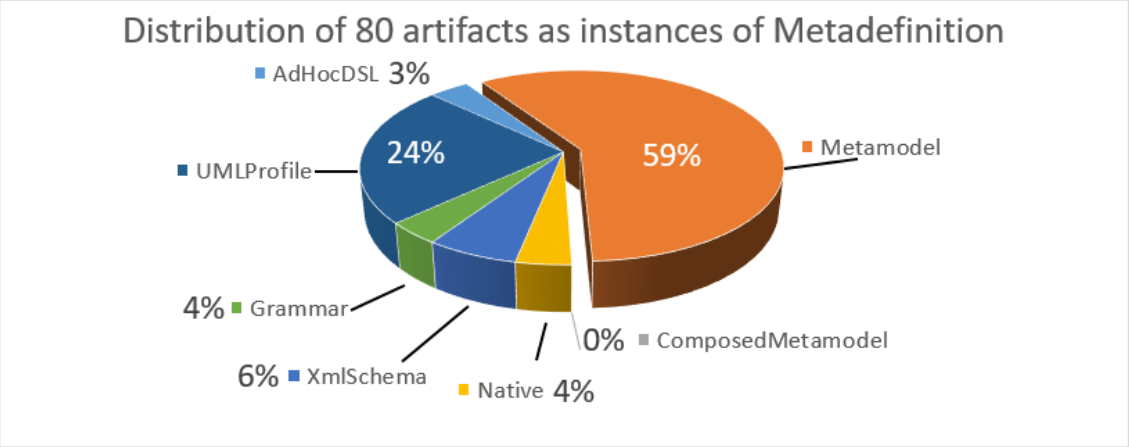


Figure 5.16: Instances of Metadefinition.

for integration is not there, asset integrators need to mine information at multiple points on the web, which is expensive and tiresome.

As demonstrated by our analysis of Figures 5.16 and 5.17, RAS ++ allows us to represent all the technical information of various types of MDE artifacts. However, we did not find in the physical artifacts (files) anything suggesting instances of `ER` and `ComposedMetamodel`, which are probably rarer cases in the tool chain.

Final Remarks

This study provides an important index of validity of RAS++. Our quantitative analysis demonstrates that metaclasses introduced into asset representations are indeed relevant. As a conclusion, we can highlight that RAS++ is representative for the third Goal from this thesis. Finally, asset providers can detail technical elements of MDE artifacts, used in tool chain approaches, which is not allowed in ReMoDD.

5.1.4 Evaluation 4 - Mining MDE Settings Hidden Data

Figure 5.18 displays the mining data describing tool dependencies through classification found in 27 of 81 assets. This information is purely descriptive and says nothing more than the name of a tool. Consequently, the asset classification data can not be mapped to representations adopted by the integration approaches/tool chain nor are they clear about the technical versions and details used by the integrator (ZAKHEIM, 2017). This is a problem that we are trying to correct with other technical representations in RAS++. In this way, the previous study allowed to select a focus group to properly evaluate the metaclasses of RAS++. Next, we describe a case study on the mining of six assets of these 27 assets in the context of

DTD	ER	ProfileGenerator	MetamodelGenerator	ASTGenerator	ProgrammingLanguage
2	0	1	6	3	1

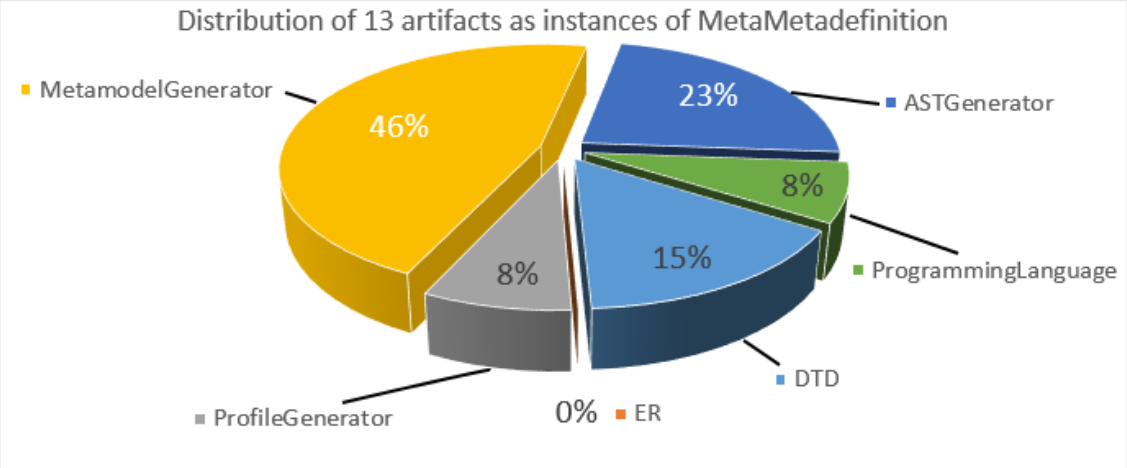


Figure 5.17: Instances of MetaMetadefinition.

Figure 5.18.

Goal

Since the previous study pointed to the existence of only one artifact of the transformation component type in the entire ReMoDD repository, our objective now is to find hidden information that suggests the relevance of the metaclasses introduced to represent transformation components: `AtomicTransformationArtifact`. This case study aims to extract implicit technical knowledge from the ReMoDD repository on MDE components. We believe that more information is possible for representation than those that we represented previously.

Research Method

We selected a sample from the first six records found in this repository. Previously we identified candidate assets that could support some sort of tool chain. Then we selected the following for mining hidden knowledge: **{A06, A07}**, using two different languages; **{A40 and A60}**, using three different languages; **{A45}** using five languages, and; **{A27}** using seven languages.

This group is now used to validate technical representations introduced in RAS++, allowing us to test the *Transformation* phase with the research questions. In this sense, we mined the hidden data from assets using the following mining steps:

1. Step 1 - Extract metadata from artifacts included in assets, representing it in assets conforming to RAS++.

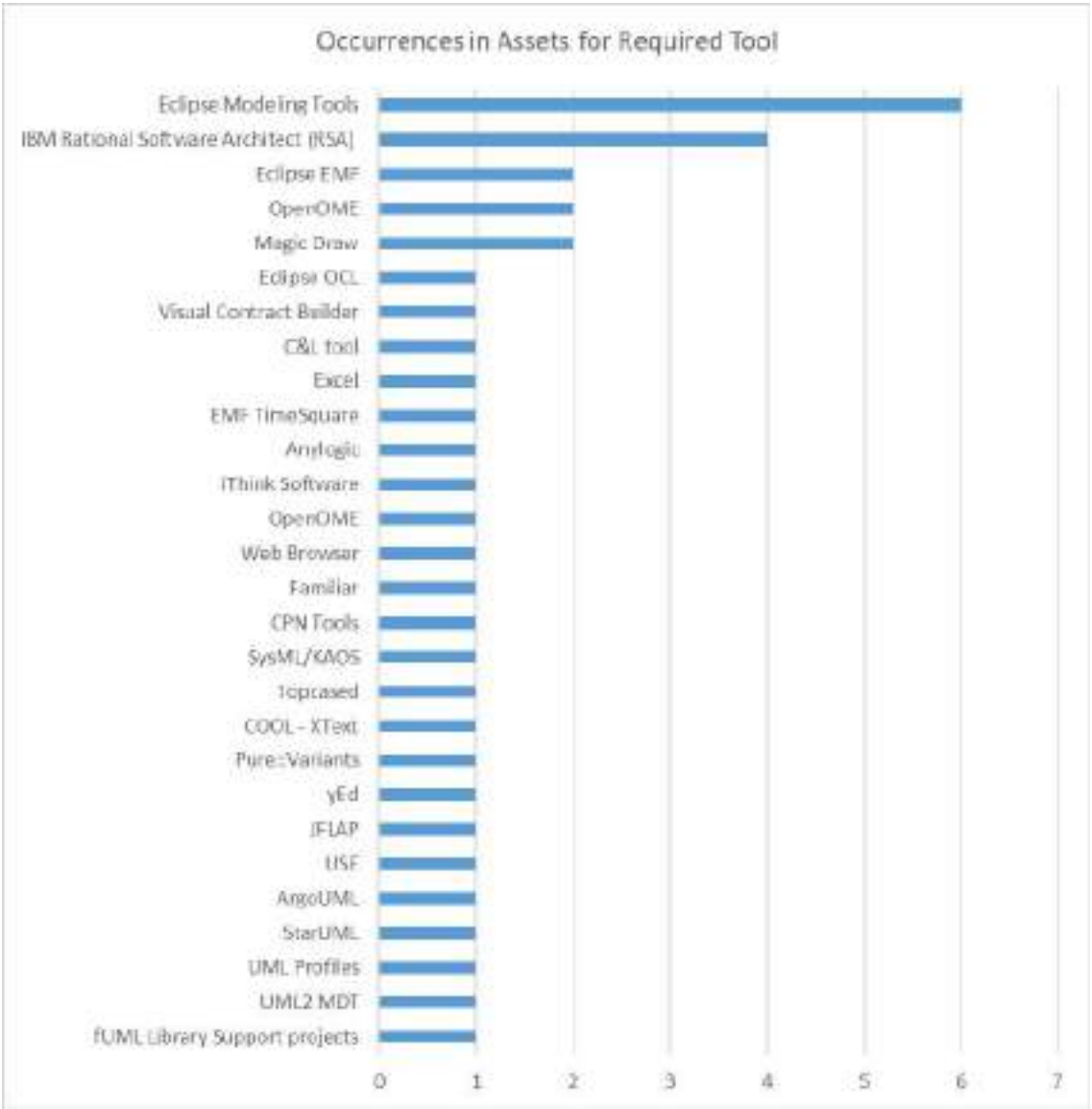


Figure 5.18: Quantitative analysis of required tools.

2. Step 2 - Extract metadata from links for tool support, if any, representing it in assets conforming to RAS++.

3. Step 3 - Since each asset in ReMoDD can be associated with a paper available in DBLP, the third step is to read the associated paper (if any), extracting metadata from settings and representing it in assets conforming to RAS++.

Analysis

Step 1

Figure 5.19 shows the result from mining the internal content of artifact *fuml-extlib-examples.zip*, which is divided in two folders: `Folder 1 - Mail` and `Folder 2 - Petstore`. It was found six models inside these folders, but with zero meta-definitions (metamodels) shared. This is due to the assumption of authors that these models are represented in UML design tools, thus in conformity with the UML2 metamodel. This assumption should be avoided when representing reusable assets to be used by inter-organizational contexts because the existence of many design tools for UML makes the exported models dependent from specific meta-definitions and exporters. For example, we found an issue for integration of these models in a tool chain: the need for specific tools.

Step 2

This issue is not easily noticed in Asset A06 because, theoretically, models are in conformity with the UML2 metamodel, thus interoperable by nature. However, after visualizing the internal content of these models, we found that those from `Folder 1` are exported differently from those available in `Folder 2`. Besides, after an extensive search on the web for the associated tool exporters, we concluded about the use of two different meta-metamodels: one used to export models in `Folder 1` conforms to UML2/MDT (EMF 2.4) and other used to export models in `Folder 2` conforms to the OMG's MOF Exporter (4.0).

Since a software engineer needs to use a specific importer tool to process each model type in a model-based task (i.e., in a tool chain), this implicit/tacit technical knowledge is always an issue for integration (BATORY *et al.*, 2013a). This knowledge should be explicit to facilitate the execution of an integration approach (ZAKHEIM, 2017). As demonstrated in Figure 5.19, a tacit knowledge, used for integration, is now explicit with the representation of Asset A06 in conformity with the RAS++ metamodel. For example, instances of `ResourceAbstraction` associate an instance of `SerializationLanguage`, shown in Figure 5.19.

Step 3

The third step is to read the paper associated with the asset to extract the information about the transformation components. This is a time-consuming step and

demonstrates the effort required today for an integrator to decide whether a given asset can be integrated into a tool chain. As shown in Figure 5.19, the execution of this step results in the representations of components shown after line 19.

The ReMoDD repository does not allow the representation of these structures. Through mining, we realized that the 27 assets, which suggested some tool chain representation through classification, in fact do. However, these representations that should be explicit are hidden in documents, which require hours of research to make a decision.

The lack of technical-level information in ReMoDD demonstrates that this repository is not well suited for tool chain needs. This could be a threat to our proposal. However, we have analyzed this limitation in ReMoDD as another reason to have a common representation language. This is because we realize through our mining that Toolboxes, associated with assets through classifications, have been shared in another repository called SHARE (GORP and GREFEN, 2012). There is no connection between these repositories today, which makes the knowledge distributed and disconnected.

RAS++ can bridge these different repositories in coopetition scenarios. In this sense, MOHAGHEGHI *et al.* (2009) claimed that establishing a bridge for the gap between technical and non-expert stakeholders is a key for success of MDE. This requires a link of the previously discussed descriptive information with technicalities. This issue is also observed by Tasktop' teams, where tool integration initiatives with software development processes (ZAKHEIM, 2017) missed qualified data for the execution of a preliminary phase for tool chain (extract hidden knowledge about tools). The author states that business stakeholders need such information, such as tool goals and purposes, business opportunities, and others; while technical stakeholders need information about tool versions, their serializations, and so on.

Final Remarks

In a coopetition scenario implemented through ReMoDD, descriptive data associated MDE Artifacts demand analysis of the internal content of shared artifacts. In other words, the information for a technical usage is not available explicitly: a tacit knowledge issue, called also as dark knowledge in (BATORY *et al.*, 2013a). For this reason, data provided in assets from ReMoDD are considered useless for automatic integration purposes. For example, in the end of preliminary phases for tool chain, this limitation in ReMoDD would make integration of MDE Artifacts a manual task. This is an issue observed in other approaches used before integration (SMOLANDER *et al.*, 2017; ZAKHEIM, 2017), thus not a particular case of ReMoDD.

Moreover, by mining the hidden knowledge from these assets, we demonstrate that RAS++ is representing data useful for automatic integration. Thus,

Figure 5.19: Technicalities from asset A06 represented with RAS++

RAS++ contributes with relevant foundations for representation in support for preliminary phases in tool chain, as recently motivated by other researchers in the area (SMOLANDER *et al.*, 2017; ZAKHEIM, 2017).

5.1.5 Evaluation 5 - Grouping Studies

This section presents the application of triangulation for the data mined from ReMoDD in previous primary studies. For instance, publications associated with ReMoDD include the following benefits for MDE artifacts: (i) support artifact use through services, (ii) facilitate artifact discovery and understanding, and (iii) foster artifact systematic reuse.

Goal

Evaluate whether these benefits are really observed through the collected data. Thus, we aim at answering the following research questions:

- *RQ 1 - Which of previous benefits are observed in ReMoDD?*
- *RQ 2 - Is ReMoDD ready for coopetition in MDE as a Service?*

Method

We applied a data triangulation considering previous studies, including a recently published one that also considers a study on criteria for quality of the descriptive data (BASSO *et al.*, 2017c).

Analysis

RQ 1 - Which of previous benefits are observed in ReMoDD?

We found that the data shared in ReMoDD lack many reuse information by business specialists and technical stakeholders. In other words, through the data provided in this repository, it is not possible to understand the business opportunities from these artifacts neither it is possible to extract the tool expert knowledge needed to build tool chains. Due to the missed information, benefits (ii) and (iii) are not noticed in practice, which answers RQ 1.

RQ 2 - Is ReMoDD ready for coopetition in MDE as a Service?

Although providing access through web services, this repository is limited in terms of representation for decision making. For this reason, ReMoDD is not ready for enabling implementation of coopetition in MDE as a Service.

This observation may explain the fall of ReMoDD, as illustrated in Figure 5.20. Between 2011 and 2014, researchers have been categorically motivated the publication of assets through important conferences in the area such as ASE, Models,

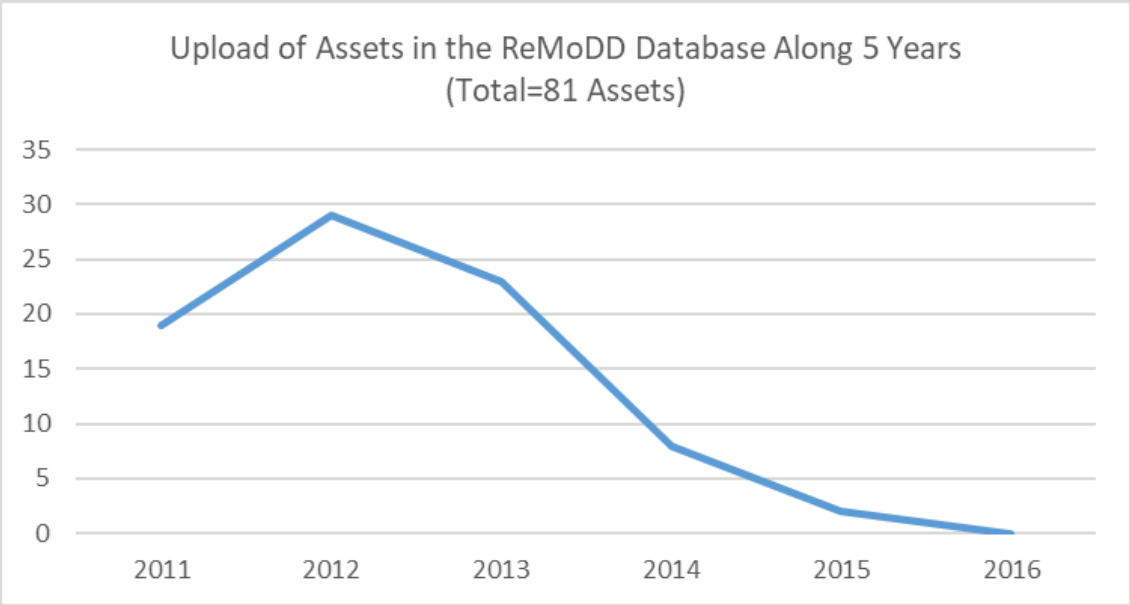


Figure 5.20: Raise and fall of the component repository ReMoDD.

ECMFA, and others. However, through the mined date from assets, we noticed a big fall of interest from 2015, until the end of adoption of ReMoDD in 2016. Thus, an open questions is: why?

Conclusion

In order to fit the needs, we therefore motivated increments in ReMoDD. MDE asset providers and consumers can benefit from representations with a better qualified descriptive data, which add semantics to such artifacts, thus promoting their adoption in global scale. Thus, to be effective, repositories such as ReMoDD needs: 1) the insertion of a more descriptive data besides keywords; 2) also provide to end-users, better structures for specification of data suggested in criteria for description of MDE Artifacts, and 3) information in technical-level that serves for automatic integration purposes.

So far, as a community that aims at promoting coopetition in the area, we made some mistakes, as can be observed by the raise and fall of the ReMoDD repository shown in Figure 5.20. It is important to notice that, although we have not found assets registered in the year 2016, ReMoDD is not dead since five new assets were registered by August, 2017. In this sense, we believe that the low quality of information found in assets shared in ReMoDD justifies its fall. Besides, the lack of interest from asset consumers may be related also with the academic intent from ReMoDD. Anyway, RAS++ is prepared to support richer data found in proposals that have been used in practices of MDE as a Service, thus introducing foundations in support for coopetition that are not observed in the ReMoDD initiative.

5.1.6 Threats to Validity

This evaluation is subject to four main threats:

Internal Validity: The trust on the correctness of the provided information for assets is an uncontrolled (not measured) factor. Besides, we have used asset data from the ReMoDD philosophy, a global repository that is feed with MDE Artifacts mostly shared for didactic purposes, not exactly for tool chain purposes. Even so, these threats do not represent issues for the validity of the performed study, since they do not affect our analysis.

External validity is about establishing the domain to which a study's findings can be generalized. A common criticism regarding case studies is the impossibility of generalizing results obtained from a single case. However, for YIN (2003), *"case studies are generalizable to theoretical propositions and not to populations or universes"*. That is to say, the results from this study represent issues related to the use of classification groups that were encountered in the repository under study, and which we therefore expect would be encountered in other sources that are similar to this one (asset repositories), in particular with respect to the hybrid nature of MDE Artifacts. Since the quality of the provided information is dependent from the asset provider, our difficulty to comprehend the shared artifacts do not represent an issue that will always occur in any asset repository built on specifications such as RAS and AMS. Furthermore, this is not a study of the full potential of a reuse repository, but a study of the practical use of representations allowed in RAS++ and how they match the needs from a global repository that actually uses assets in its reuse mechanisms.

Construct validity: This threat relates to representation of assets for non-realistic scenarios, or data provided by the researcher, or data considering a unique software development context. It makes the representation of assets less reliable. This is not an issue for this study since assets found in ReMoDD are from realistic scenarios, elaborated by several asset providers, from different software development contexts.

Conclusion validity: A low statistical power threat is included when the low number of population is collected. This is not our case because we mined 81 assets, thus an expressive number that characterizes an adequate statistical power. Due to the low diversity in representations for federation is not considered (i.e., all the federation is internal to ReMoDD), this could introduce noise in our conclusions. For this reason, we have not included in our conclusions statistical analysis for federation. Regarding confidentiality threats, the literature of the area on reporting the development of MDE Artifacts usually considers these elements confidential. This is not an issue for this study since there is no confidentiality in the assets

shared in ReMoDD. The employed statistical methods and sample size choices are adequate to evaluate the representativeness of RAS++. Thus, the focus group of assets provides valid data for drawing conclusions.

5.2 Combinatorial Proof

Previous studies presented a focus group of languages used in tool chain scenarios. In order to demonstrate the relevance of a pivotal representation language for MDE as a Service, this section extends previous analytical comparisons with a combinatorial proof by analyzing the development effort of connectors that should be used in the *Transformation* phase.

We analyzed three sets of representations that could be introduced in a coopetition scenario for MDE as a Service shown in Figure 5.21. The integration of these representations at the end of the *Transformation* phase implies on the development of connectors. Our intent is to simulate temporal data, i.e., quantifying implementation possibilities found in the state of the art, by measuring the effort for development of connectors through an analysis of three sets of data:

1. Analysis from data collected in an ad-hoc literature review (BASSO *et al.*, 2013a), which reports approaches for tool chain proposed between 2005 and 2012.

2. Analysis of data between 2013 and march 2015, collected in a literature mapping for the qualification exam, available at `prisma.cos.ufrj.br/wct/eqmap1.pf`.

3. Analysis for the future, using data extracted from three of our recent mapping studies shown in Table 5.9.

Table 5.9: Mapping studies for RAS++ conception

Id	Title	Available at
M01	Semantic Properties of Software Components	`prisma.cos.ufrj.br/wct/ms02.pdf`
M02	Intents from Asset Platforms and Their Properties	`prisma.cos.ufrj.br/wct/ms03.pdf`
M03	MDE Settings Intents and Their Properties	`prisma.cos.ufrj.br/wct/ms04.pdf`

This study considers the relevance of RAS++ in support for all the three phases discussed previously. Every one would need a sort of conversion from a RAS++ model (see Figure 5.21.3) to a target representation: For a tool chain, as illustrates Figure 5.21.2, or for an asset repository illustrated in Figure 5.21.1.

We opted for the term “converter” or “conversion” for a general application of generative techniques, rather than “transformation” that is more restrictive for a phase with the same name. Likewise, a converter can be programmed: as a black-box Java class that queries a database and generates a target representation; as a

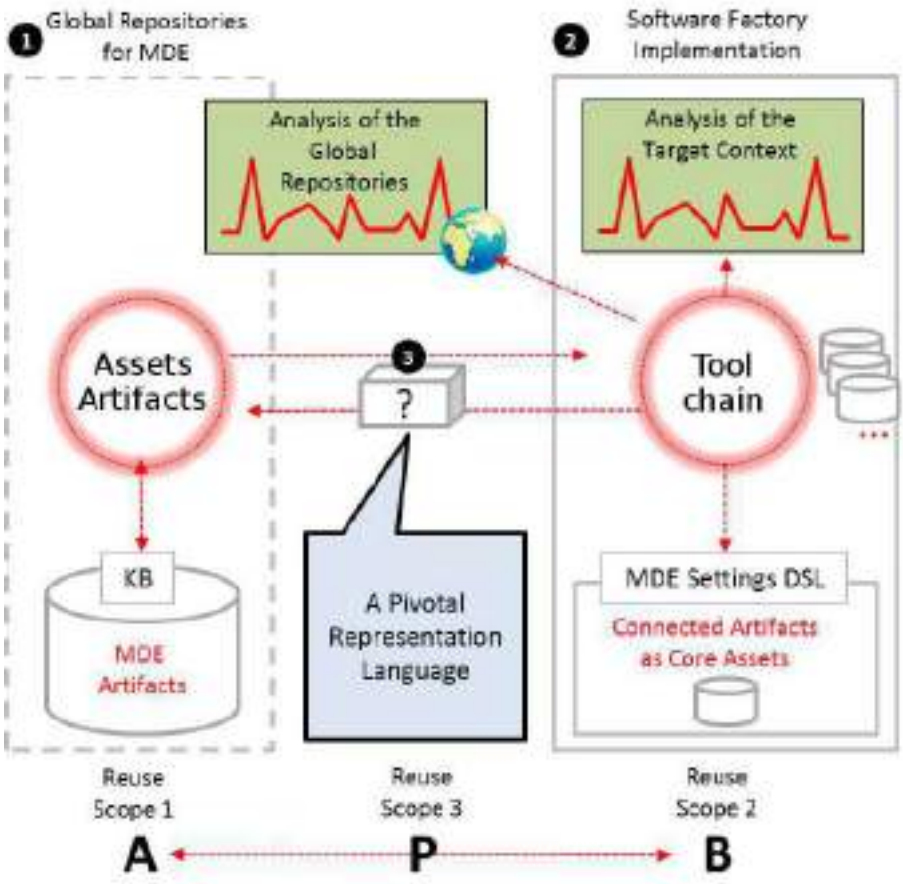


Figure 5.21: Desirable scenario for coopetition in MDE as a Service.

model-to-model transformation developed with ATL (BÉZIVIN, 2005); as a binding for reuse of ATL code developed in Bentõ DSL (CUADRADO *et al.*, 2014) or a framework such as T-Core (SYRIANI *et al.*, 2015), as a dynamic content generator with Velocity templates, etc. This definition includes, therefore, all possible representations found in a coopetition scenario from Figure 5.21.

This study is organized as follows. Section 5.2.1 presents our study goal and Section 5.2.2 the adopted research method. Section 5.2.3 depicts the analysis and Section 5.2.4 draws conclusions.

5.2.1 Goal

It is not our intent the development of connectors, but due to the absence of a common representation language, we need to understand the scale of costs for implementation of coopetition in MDE as a Service through a quantification of connectors. Due to this diversity of languages, we believe that without a common representation language, the development of connectors for the complex motivated scenario would be impracticable. However, we still do not acknowledge how impracticable or

complex it is, thus characterizing an interesting topic for investigation.

As shown in Table 5.10, our goal is to find out numbers that lead us to conclude whether RAS++ is important for reducing this complexity that we are prospecting for integration. In this sense, we aim at answering the research question: **Q7: Which is the effort required to implement coopetition in the context of MDE as a Service?** This question is divided in two sub-questions: *Q7.1: Which is the current effort?* and *Q7.2: Which is the effort with RAS++?*

Table 5.10: Mathematical evaluation goal of the motivated scenario

Analyze	The amount of DSLs introduced after 2015 in support for integration
For the purpose of	Characterizing
Regarding	Scenarios for coopetition in MDE as a Service
From the viewpoint of	Researcher
In the context of	Development costs for connectors.

5.2.2 Research Method

We first applied a data triangulation from literature mappings, diving representations in three groups. Our first impression is that, as more languages are introduced into the problem, more effort is needed for the development of connectors. After searching the literature on machine translation (AIGNER and ZIEGLER, 1998), we find out a mathematical answer that describes this scenario: a quadratic combinatorial explosion for the development of connectors. It is an equation that describes this issue, as follows:

$$Quadratic, C = \frac{N(N-1)}{2}$$

In this sense, let `C` be the number of possible converters and `N` be the number of representation languages (DSLs for MDE Settings and/or repository metamodels) found in the scenario shown in Figure 5.21. Through this formula, it is possible to reach the effort needed for integration between all the possible representations inserted into a coopetition scenario. This equation allows to formulate a premise: following the current state of the art of research that does not present a pivotal language, software engineers would fall in the issue of the quadratic combinatorial explosion.

For pivot language, we used the following definition:

> *"A pivot language, sometimes also called a bridge language, is an artificial or natural language used as an intermediary language for translation between many different languages - to translate between any pair*

of languages "A" and "B", one translates "A" to the pivot language "P", then from "P" to "B". Using a pivot language avoids the combinatorial explosion of having translators across every combination of the supported languages, as the number of combinations of language is linear, rather than quadratic - one need only know the language "A" and the pivot language "P" (and someone else the language "B" and the pivot "P"), rather than needing a different translator for every possible combination of "A" and "B". " (KRIPPENDORFF, 2010)

In order to confirm this premise, we extracted proposals for representation languages. Then, we have applied an analysis of the number of connectors required in each moment of this research, demonstrating this combinatorial issue. Likewise, the literature states that it is possible to avoid combinatorial issues through a pivotal language, reducing the combinations for a linear problem described by the equation bellow.

$$Linear, C = N - 1$$

In the next sections, we apply these equations to analyze the effort for integrating a group of representations.

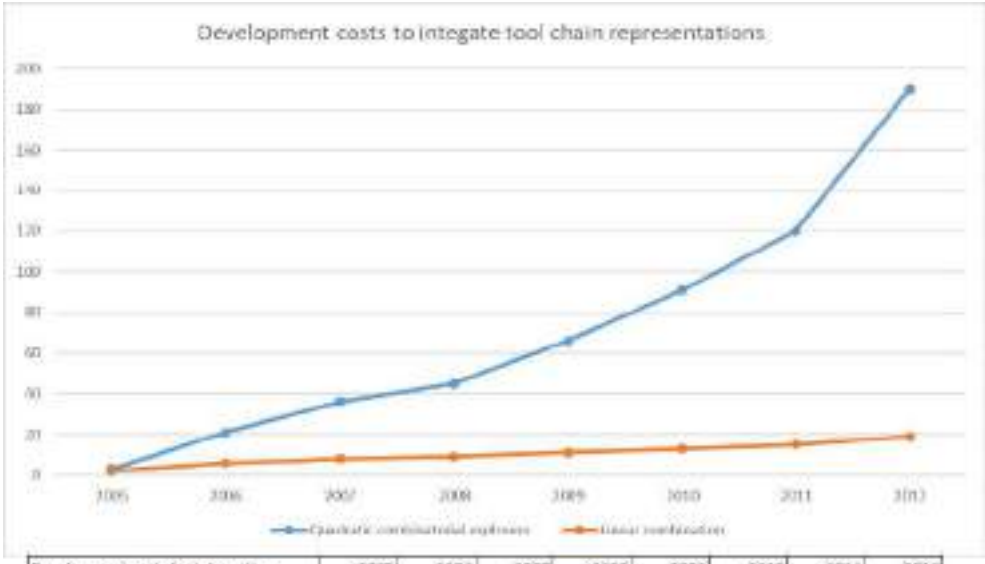
5.2.3 Analysis

In our analysis of data from ad-hoc literature review, we considered 20 representations for tool chain, as shown in Table 5.11. These representations are classified in reuse scope 2, as shown in Figure 5.21, thus scoping some tool mechanism in support for tool chain integration.

Table 5.11: Ad-hoc mapping of toolboxes for tool chain before 2012

Id	Paper	Year
S01	From the workfloor: Developing workflow for the generative model transformer.	2005
S02	Integrating platform selection rules in the model driven architecture approach.	2005
S03	Platform-independent modelling in mda: Supporting abstract platforms.	2005
S04	Blackbox composition of model transformations using domain-specific modelling languages.	2006
S05	A framework for transformation chain development processes.	2006
S06	Towards a transformation chain modeling language.	2006
S07	Using the fomda approach to support object-oriented real-time systems development.	2006
S08	Handling variability in model transformations and generators.	2007
S09	On model typing	2007
S10	MDA Tool Components: a proposal for packaging know-how in model driven development	2009
S11	Feature-oriented refinement of models, metamodels and model transformations.	2009
S12	Incremental development of model transformation chains using auto mated testing.	2009
S13	Combining independent model transformations.	2010
S14	Domain-specific composition of model deltas.	2010
S15	Generic model transformations: Write once, reuse everywhere.	2011
S16	Simplifying model transformation chains by rule composition.	2011
S17	Realizing model transformation chain interoperability.	2012
S18	Automatic adaptation of transformations based on type graph with multiplicity.	2012
S19	Chaining model transformations.	2012
S20	Using feature models to tame the complexity of model transformation engineering.	2012

Figure 5.22 shows a graph comparing development effort for connectors, by year, before 2012. For the year 2005, we found just three proposals for representation of



Development costs for integration	2005	2006	2007	2008	2009	2010	2011	2012
Publications by year	3	4	2	1	1	2	2	4
Sum of languages	3	7	9	10	12	14	16	20
Quadratic combinatorial explosion	3	21	36	45	66	91	120	190
Linear combination	2	6	8	9	11	13	15	19

Figure 5.22: Development effort by year for connectors before 2012.

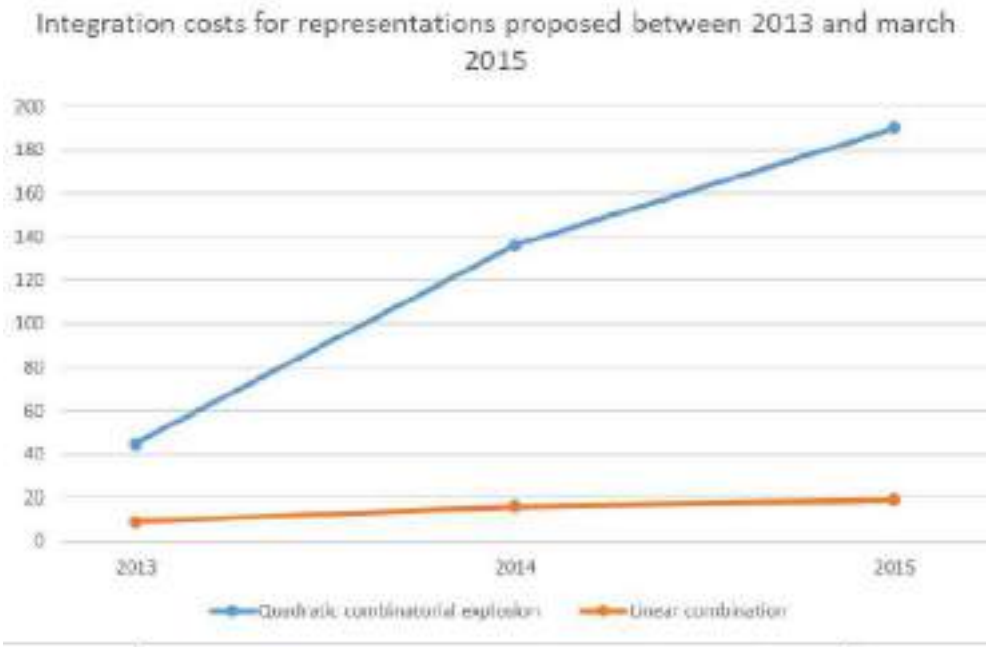
MDE Settings. For this reason, differences between a combinatorial explosion and a linear combinatorial solution are not evident. Along the years, more representations are proposed, thus increasing the effort for integration since each representation must be combined with each other.

Table 5.12: Mapping of toolboxes for tool chain after 2012

Id	Paper	Year
S21	On the Complex Nature of MDE Evolution	2013
S22	Typing artifacts in megamodeling	2013
S23	Engineering model transformations with transML	2013
S24	Supporting Large Scale Model Transformation Reuse	2013
S25	MTC Flow: A Tool to Design, Develop and Deploy Model Transformation Chains	2013
S26	Localized model transformations for building large-scale transformations	2013
S27	A Common Representation for Reuse Assistants	2013
S28	MTP: Model Transformation Profile	2013
S29	Genericity for model management operations	2013
S30	Semantic Conflicts Detection in Model-driven Engineering	2013
S31	A Component Model for Model Transformations	2014
S32	On the modeling and generation of service-oriented tool chains	2014
S33	Towards Facilities to Introduce Solutions for MDE in Development Environments with Reusable Assets	2014
S34	Dealing with Traceability in the MDD of Model Transformations	2014
S35	Adapting transformations to metamodel changes via external transformation composition	2014
S36	Extending JUnit 4 with Java Annotations and Reflection to Test Variant Model Transformation Assets	2014
S37	Generative Adaptation of Model Transformation Assets: Experiences, Lessons and Drawbacks	2014
S38	A survey of approaches for verifying model transformations	2015
S39	T-Core: a framework for custom-built model transformation engines	2015
S40	Reusable Model Transformation Components with bentō	2015

Table 5.12 shows 20 more references found between 2013 and march 2015, also classified in reuse scope 2 of Figure 5.21. The difference is that these ones are more modern. Figure 5.23 presents an analysis of development effort by year, considering only these recent proposals and ignoring the past representations for MDE Settings. This way, 20 new representations are found in the current scenario for coopetition, which means that some compete and other cooperate in a MDE-based process.

Table 5.13 presents 29 papers proposing a sort of representations classified in



Development costs for integration	2013	2014	2015
Publications by year	10	7	3
Sum of languages	10	17	20
Quadratic combinatorial explosion	45	136	190
Linear combination	9	16	19

Figure 5.23: Development effort by year for connectors from 2013 until march 2015.

reuse scope 1 of Figure 5.21, from 2005 until the present, which considered the storage of some MDE Artifact in assets, repositories or platforms. Figure 5.24 shows that, even if we ignore all the previous representations languages, a combinatorial explosion issue is also observed in coopetition of MDE Artifacts shared by these infrastructures. In this case, we are ignoring whether the infrastructure can represent MDE Settings or not, as demand the previous representations.

Figure 5.25 presents our analysis of effort for integration of all these representations in a coopetition scenario.

Q7.1: Which is the current effort?

Currently, we would need a total of 2.346 connectors for integration performed without a common representation language.

Q7.2: Which is the effort with RAS++?

The same coopetition scenario implemented with RAS++ would demand 68 connectors. Through RAS++, Software Engineers are now allowed to integrate these and other representations with less effort for development of converters. Likewise, they still need to develop these converters, but in the order of `N-1` instead of `(N*(N-1))/2`. Therefore, through a combinatorial proof, we found a new index suggesting a benefit associated with RAS++: it reduces effort in integration.

Development costs for integration	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016
Publications by year	1	1	2	1	3	1	0	3	6	5	3	3
Sum of languages	1	2	4	5	8	9	9	12	18	23	26	29
Quadratic combinatorial explosion	0	1	6	10	28	36	36	66	153	253	325	406
Linear combination	0	1	3	4	7	8	8	11	17	22	25	28

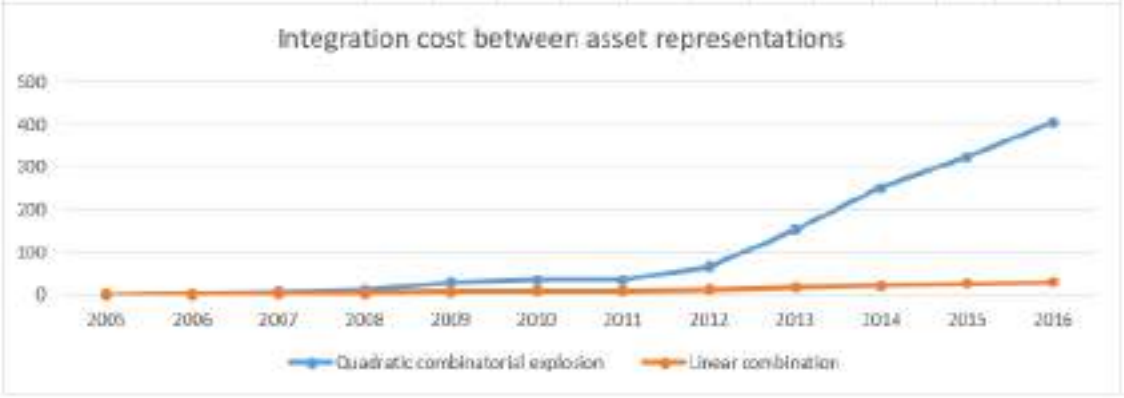


Figure 5.24: Development effort by year for asset connectors.

Table 5.13: Mapping of infrastructures for assets

Id	Paper	Year
S41	RAS Reusable Asset Specification Version 2.2 November 2005	2005
S42	X-ARM: an asset representation model for component repository systems.	2006
S43	Extending reusable asset specification to improve software reuse.	2007
S44	Repository for Model Driven Development (ReMoDD).	2007
S45	A Representation Model for Reusable Assets to Support User Context.	2008
S46	Reusable SOA Assets Identification Using E-Business Patterns.	2009
S47	Design and Implementation of RAS-Based Open Source Software Repository	2009
S48	Asset Management Specification.	2009
S49	Analyzing the Concept of Components in the Brechó-VCM Approach through a Sociotechnical and Software Reuse Management Perspective.	2010
S50	A Catalogue of Refactorings for Model-to-Model Transformations	2012
S51	A Light-weight Tool Integration Approach - From a Tool Integration Model to OSLC Integration Services.	2012
S52	Towards Tool Integration through Artifacts and Roles	2012
S53	Supporting the internet-based evaluation of research software with cloud infrastructure	2012
S54	Typing artifacts in megamodeling	2013
S55	Multi Back-Ends for a Model Library Abstraction Layer	2013
S56	Establishing tool chains above the service cloud with integration models	2013
S57	Integrating Modeling Tools in the Development Lifecycle with OSLC: A Case Study	2013
S58	A Common Representation for Reuse Assistants	2013
S59	Software Ecosystems Comprehension and Evolution	2013
S60	Globalizing Modeling Languages	2014
S61	Model transformation intents and their properties	2014
S62	Automated Chaining of Model Transformations with Incompatible Metamodels	2014
S63	MDEForge: An extensible Web-based modeling platform	2014
S64	Modeling of tool integration resources with OSLC support	2014
S65	Collaborative Repositories in Model-Driven Engineering	2015
S66	Enabling the reuse of stored model transformations through annotations	2015
S67	Tool Integration by Models, Not Only by Metamodels - Applying Modeling to Tool Integration	2015
S68	Software component decision-making: In-house, OSS, COTS or outsourcing - A systematic literature review	2015
S69	The importance of socio-technical resources for software ecosystems management	2015

5.2.4 Conclusion

Our analysis of the state of the art shows that, if coopetition in MDE as a Service is indeed something to be implemented in future, this is the right moment to build foundations in support. For instance, since other IT market segments have reported an increase in business (PALMQUIST, 2014), we are assuming that asset coopetition could benefit MDE practitioners as well, extending initiatives for MDE as a Service (BRAMBILLA and FRATERNALI, 2014; BRIAND *et al.*, 2012; MONTEIRO *et al.*, 2014a,b) towards global cooperation. In this sense, the advent of a pivotal language would simplify this cooperation by, for example, promoting an

Development costs for integration	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016
Publications by year	4	5	4	2	5	3	2	7	16	12	6	3
Sum of languages	1	9	13	15	20	23	25	32	48	60	66	69
Quadratic combinatorial explosion	0	36	78	105	190	253	300	496	1128	1770	2145	2346
Linear combination	0	8	12	14	19	22	24	31	47	59	65	68

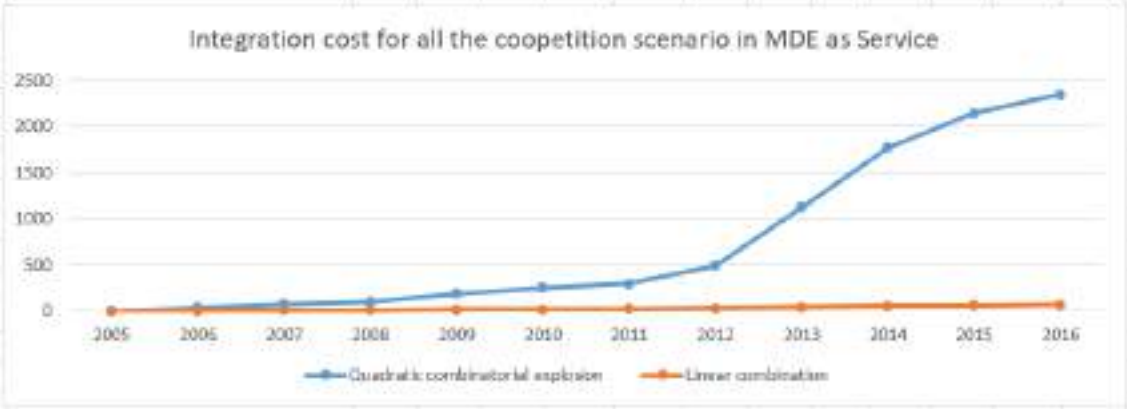


Figure 5.25: Development effort by year for all the connectors.

interchange of data represented with FOMDA DSL (BASSO *et al.*, 2013a), which is focused in orchestration and adaptation of model transformation components to other DSLs, such as TIL (BIEHL *et al.*, 2014), focused in MDE tool integration features.

For some reasons that are still not clear, the state of the art and practice are incapable of implementing coopetition in the large. We found that the complexity of this scenario and the lack of a common representation language are some of these reasons. Considering possibilities found in the scenario shown in Figure 5.21, existing representation languages are limited to be elected as pivotal. The selection of one or other would imply in loss of information across conversions, which is called a bad pivotal choice (KRIPPENDORFF, 2010). Thus, as more representative a language is, the more qualified it is for conversions.

This study suggests that this combinatorial issue should be discussed by the literature of the area. For example, so far, Software Engineers have presented benefits that could be promoted through modern repositories (COMBEMALE *et al.*, 2014; CORREIA *et al.*, 2016; LIMA *et al.*, 2016; ROCCO *et al.*, 2015; SANTOS *et al.*, 2016). However, considering the full scenario (big picture), they have been neglecting the effort needed by end-users to integrate shared artifacts in target contexts. This is because, at least in Software Engineering, words such as collaboration, cooperation and competition demand the management of assets external to a repository (i.e., federation) (AXELSSON *et al.*, 2014; BADAMPUDI *et al.*, 2016). Connections between different infrastructures for assets imply in an integration issue (ZAKHEIM, 2017), thus making the term “pivotal” a required discussion topic.

As a contribution to this research area, we presented such a discussion. In this

study, we considered an analysis of this effort, but limited to tool chain contexts. This means that other contexts for integration should be discussed in the future as well. Our conclusion is that, currently, the state of the art presents an issue in the form of a quadratic explosion combination that probably is present in other integration contexts for coopetition: it increases year after year, thus becoming unmanageable in a long-term perspective, so this issue should be explored right now.

Our analysis also suggests a relationship between effort for integration and the number of representations. For example, we found that without a common representation, the development of a new repository adds more complexity for integration, thus making coopetition harder to implement along the years. This is somehow paradoxical because the current research is proposing more repositories and tool chain representations as solutions. However, this is also making integration harder. As consequence, due to the large amount of representations, coopetiton in MDE as a Service is becoming less feasible along the years. Thus, as a possible solution, pivotal languages like RAS++, AMS and RAS tend to bridge these representations in a linear combination rather than quadratic.

Finally, it is important to mention that RAS++ is a pivotal representation language. This way, a last conclusion through the analyzed data is an increase of the feasibility for implementation of competition in MDE as a Service, which is important to find the basis and promote initiatives in this direction. Due to costs of developing each connector, the analyzed data suggest that the implementation of coopetition is too much expensive to take the risk when performed without an adequate pivot language. Since the development would consume 2,9% of the effort for connection of 69 representations (68 in comparison to 2.346), integrations built on RAS++ are more feasible for implementation. Thus, RAS++ is important to foster coopetition initiatives in the future.

5.3 Comparison Studies

This section presents comparative analytical studies evaluating the quality attribute "representativeness" from RAS++. In order to highlight representativeness from RAS++, we present 10 property tables comparing competing toolboxes within some representational focus. These studies consider, therefore, a comparison of properties introduced for Specification and Transformation phases in RAS++ with other approaches that could be selected as pivotal for the development of connectors.

This study is organized as follows. Section 5.3.1 presents our study goal and Section 5.3.2 the adopted research method. Section 5.3.3 depicts the analysis and Section 5.3.4 draws conclusions.

5.3.1 Goal

Our goal is to compare properties found in support for assets, MDE Artifacts and Settings, through the following research question: **Q8: Is RAS++ representative to play the role of a common language for tool chain and assets in the context of MDE as a Service?**

Understanding properties from hybrid toolboxes is an important requirement for evaluation of the quality attribute "representativeness" of RAS++.

5.3.2 Research Method

In order to evaluate representativeness, we selected 12 representations for tool chain, shown as papers in Table 5.14, and six representations for assets shown Table 5.15. These papers are derived from four recent mapping studies shown in Table 5.16. Based on these papers, it was constructed 10 property tables with structural features from these representations. These property tables demonstrate complementarity and overlapping in toolboxes that are inserted in the context of RAS++. For example, in the mapping `M03`, we characterized these toolboxes in different intents for MDE as a Service, so that they can be inserted as competitors and cooperators in different scenarios. Therefore, since these properties are discovered in four primary mapping studies, in this study, we just applied them as a template for comparisons.

Table 5.14: Studies in tool chain used for comparison with RAS++

Study	Paper	Title	Year
TC01	(GUOJIE and BAOLIN, 2009)	An Encapsulation Structure and Description Specification for Application Level Software Components	2009
TC02	(PALUDO *et al.*, 2011)	Applying pattern structures to document and reuse components in component-based software engineering environments	2011
TC03	(LÚCIO *et al.*, 2014)	Model transformation intents and their properties	2014
TC04	(CRIADO *et al.*, 2015)	Enabling the reuse of stored model transformations through annotations	2015
TC05	(BIEHL *et al.*, 2014)	On the modeling and generation of service-oriented tool chains	2014
TC06	(ZHANG and MOLLER-PEDERSEN, 2014)	Modeling of tool integration resources with OSLC support	2014
TC07	(VIGNAGA *et al.*, 2013)	Typing artifacts in megamodeling	2013
TC08	(GARCÉS *et al.*, 2014)	Adapting transformations to metamodel changes via external transformation composition	2014
TC09	(YIE *et al.*, 2012)	Realizing Model Transformation Chain interoperability	2012
TC10	(ETIEN *et al.*, 2012)	Chaining model transformations	2012
TC11	(BASSO *et al.*, 2014e)	Generative Adaptation of Model Transformation Assets: Experiences, Lessons and Drawbacks	2014
TC12	(CUADRADO *et al.*, 2014)	A Component Model for Model Transformations	2014

Table 5.15: Compared asset specifications/repositories

Study	Documentation	Year
A01	Reusable Asset Specification Version 2.2 Av. at www.omg.org/spec/RAS/	2005
A02	Asset Management Specification. Av. at open-services.net/wiki/asset-management/	2016
A03	ReMoDD - The Repository for Model Driven Development Av. at http://www.cs.colostate.edu/remodd/v1/repository/	2016
A04	MDE Forge Av. at http://www.mdeforge.org/	2017
A05	SEMAT (Software Engineering Method and Theory) Av. at http://semat.org/ and Essence (OMG, 2015)	2017
A06	ReuseECOS: An Approach to Support Global Software Development Through Software Ecosystems (SANTOS and WERNER, 2012)	2017

5.3.3 Analysis

Properties introduced in RAS++ are relevant in specific scenarios for coopetition in MDE as a Service. This means that some properties can be relevant for representations found in one scenario and irrelevant for others. This section presents six scenarios for comparison, each one introducing different properties: 1-Asset Specifications; 2-Component Representations; 3-OO and Database Representations; 4-Tool Chain Representations; 5-Pivotal Representations, and 6-Software Process Integration.

Scenario 1-Asset Specifications

Table 5.16: Mapping studies with properties from MDE Artifacts and Settings

Id	Title	Available at
M01	Semantic Properties of Software Components	prisma.cos.ufrj.br/wct/ms02.pdf
M02	Intents from Asset Platforms and Their Properties	prisma.cos.ufrj.br/wct/ms03.pdf
M03	MDE Settings Intents and Their Properties	prisma.cos.ufrj.br/wct/ms04.pdf
M04	Diversity of MDE Toolboxes and Their Uncommon Properties	prisma.cos.ufrj.br/wct/ms05.pdf

Table 5.17 compares toolboxes/representations for assets described in Table 5.15. In order to fit the needs in MDE, as well as modernize RAS to novelties available in AMS (e.g., artifact federation), RAS++ presents considerable contributions to bridge all these specifications and repositories recently proposed for adoption.

Scenario 2-Component Representations

General Software Components

Table 5.18 shows RAS++ in comparison to approaches for software component classifications. It is possible to notice that different approaches emphasize different needs for representation. For example, TC01 emphasizes representation of technicalities for components, while TC02 emphasizes the quality for descriptive information

Table 5.17: Property table 1: asset representations

LEGEND [Distributed=D, Local=L], [SECO = Software Ecosystem]

Property	A01	A02	A03	A04	A05	A06	RAS++
1. Is/Use Some Specification	Yes	Yes	No	No	Yes	No	Yes
2. Represent Any Artifact	Yes	Yes	Yes	Yes	No	Yes	Yes
3. Instructive Reuse Data	Yes	No	No	No	No	Yes	Yes
4. Artifact Federation	L	D	L	D	L	D	D
5. Light-weight Extensibility	No	No	No	Yes	No	No	Yes
6. Asset Profile	Yes	No	No	No	No	No	Yes
7. Descriptor group	Yes	No	Yes	No	No	Yes	Yes
8. Classification	Yes	Yes	Yes	Yes	Yes	Yes	Yes
9. Free form	Yes	Yes	Yes	Yes	Yes	Yes	Yes
10. Context	Yes	No	No	No	Yes	Yes	Yes
11. License	Yes	No	No	No	No	Yes	Yes
12. User feedback	No	No	Yes	No	Yes	Yes	Yes
13. Represent some MDE Settings	No	No	No	Yes	Yes	No	Yes
14. OCL Rules in Metamodel-Level	No	No	No	No	Yes	No	Yes
15. Web Services Access	Yes	Yes	No	Yes	No	Yes	Yes
16. SECO perspectives	No	No	No	No	No	Yes	Yes

Table 5.18: Property table 2: software component assets

LEGEND [Variability Point=VP] [Create, Read, Update, Delete = CRUD]

Property	TC01	TC02	RAS	ReMoDD	RAS++
1. Interface specification	Yes	Yes	Yes	No	Yes
2. Datatype	Yes	No	No	No	Yes
3. IO constraints	Yes	No	No	No	Yes
4. Component chain	Yes	No	No	No	Yes
5. Form/CRUD specificity	Yes	Yes	No	No	No
6. Light-weight Extensibility	No	No	No	No	Yes
7. Quality for descriptive information	No	Yes	No	No	Yes
8. Artifact VP & VP Binding	Yes	Yes	Yes	No	Yes
9. Asset VP & VP Binding	No	No	No	No	Yes

through structures that organize decision making features. Meanwhile, RAS++ considers both perspectives for representation, including criteria for quality in descriptive information as an ongoing work for the Acquisition phase (BASSO *et al.*, 2017c). Therefore, RAS++ is representative for coopetition scenarios for software components.

Model Transformation Intents

In a similar way, software components demand appropriate classifications, some contributions for model transformation intents have emerged as a way to classify these MDE Artifacts in repositories with standard taxonomy too. **Model Transformation Intents (MTI)**, allowing a classification of some MDE resources through properties for taxonomy in the area. These works are related to repositories and aim at promoting the collaboration in MDE through the globalization of MDE Artifacts. Thus, similar to software component reuse through assets, the focus of MTI is in descriptive information.

Table 5.19 shows two approaches for MTI. In comparison to CRIADO *et al.* (2015), which proposed structures for classification of modules/pieces that com-

Table 5.19: Property table 3: model transformation intents (MTI)

LEGEND [Detailed=D], [Flexible=F], [S=Superficial]

Property	TC03	TC04	RAS++
1. Semantics for MDA Views	Yes	Yes	Yes
2. Classification of MT Intentions	Yes	No	Yes
3. Transformation Properties	S	D	D + F
4. Consistency in Semantics	No	No	No
5. MDE Artifacts & Settings	S	S	D
6. Structured Descriptive Data	S	S	D
7. Light-weight Extensibility	No	No	Yes
8. Scenario 1	No	No	Yes

pose a transformation (different rules) with annotations, RAS++ is limited to the representation of instances of `TransformationArtifactFrag`. However, through extensibility mechanisms, it is possible to represent annotated pieces of a transformation as well. Thus, we conclude that RAS++ is representative for pivoting MTI approaches.

Scenario 3-OO and Database Representations

RAS++ can be characterized as an Architectural Description Language (ADL) for Object-Oriented (OO) and Relational Database Perspective (RDP) representations. It was not our initial intent to introduce a new ADL for OO and RDP. However, we found contributions from ADLs for O.O. and RDP in tool chain approaches, which required descriptions for MDE Artifacts with properties from these two development concepts. As suggests our mining of ReMoDD repository, the distinction of components built in support for one or other development paradigm is important for decision making. Thus, this is the reason why we inserted these properties in RAS++.

The result is an hybrid representation for these two distinct concepts, as shown in a comparison from Table 5.20. RAS++ matches properties from some of the more important representations used in four toolboxes presented in Table 5.21. We selected UML, Ecore, JDT, BrM (Conceptual, Logic and Physic models) and RDP (only Physic model) for demonstration because they show how hybrid a tool chain can be in terms of artifact typing. For example, when analyzing RDP, we considered all the reported toolboxes discussed by BATORY *et al.* (2013b). In this paper, Batory reports success on the use of RDP with MDE, where model transformations are built on Java programs and database queries, thus introducing RDP in the context of tool chain integration. This demonstrates that artifact typing in RAS++ is relevant for representations in whatever is considered in the literature as "models". Therefore, by using RAS++, one can use OO or RDP properties in tool chains and, additionally, properties from MDE Artifact Types.

Table 5.20: Property table 4: Hybrid properties from representations for OO and RDP

Property	UML	Ecore	JDT	BrM	RDP	RAS++
1. Package (OO)	Yes	Yes	Yes	No	Yes	Yes
2. Interface (OO)	Yes	Yes	Yes	No	Yes	Yes
3. Datatype (OO and RDP)	Yes	Yes	Yes	Yes	Yes	Yes
4. Operation (OO)	Yes	Yes	Yes	No	Yes	Yes
5. Class (OO)	Yes	Yes	Yes	No	Yes	Yes
6. Property (OO and RDP)	Yes	Yes	Yes	Yes	Yes	Yes
7. Table (RDP)	No	No	No	Yes	Yes	Yes
8. View (RDP)	No	No	No	Yes	Yes	Yes
9. Index and Constraints (RDP)	No	No	No	Yes	Yes	Yes
10. Enumeration (OO and RDP)	No	Yes	Yes	No	Yes	Yes
11. Inheritance (OO and RDP)	Yes	Yes	Yes	Yes	Yes	Yes
12. Realization (OO)	Yes	Yes	Yes	No	Yes	Yes
13. Dependency (OO)	Yes	Yes	No	No	No	Yes
14. Association (OO and RDP)	Yes	Yes	No	Yes	Yes	Yes

Table 5.21: Hybrid toolboxes used for comparison with RAS++

Study	Paper	Title	Year
UML	`www.omg.org/spec/UML/2.5/`	Unified Modeling Language Version 2.5	2015
Ecore	`www.eclipse.org/ecoretools/`	Eclipse Ecore Tools	2017
JDT	`www.eclipse.org/jdt/`	Eclipse Java development tools (JDT)	2017
BrM	`brmodeloweb.ufsc.br`	BR-Modelo Web	2017
RDP	(BATORY *et al.*, 2013b)	Teaching Model Driven Engineering from a Relational Database Perspective	2013

It is important to notice that these hybrid properties are not only important for tool chain representations. For example, an ongoing work is replacing UML from the core implementation in RDL tool support (LUCAS *et al.*, 2017; OLIVEIRA *et al.*, 2011). This is allowing us to apply reuse processes independently from the representation language adopted for representation of Object Oriented Frameworks (OOF). Through this study, our goal is to simplify the reuse scenario in OOF Instantiation (or OOFI). This is possible by using RAS++ as the core representation in Reuse Tool (OLIVEIRA *et al.*, 2011), and by applying converters from RAS++ to other representations such as UML, Ecore, Eclipse JDT, and others.

Scenario 4-Tool Chain Representations

Since tool chain representation languages are built on some of the previous properties, this section makes use of previous Scenarios to demonstrate some hybrid approaches for tool chain. In the following, we compare RAS++ with the state of the art resultant from literature reviews for tool chain approaches, as shown in Table 5.14.

Model-Driven Service Instantiation (MDSI)

The scenario for MDE adoption is heterogeneous and requires the use of several tools not based on models. Model-Driven Service Integration (MDSI) classi-

Table 5.22: Property table 5: Model-Driven Service Instantiation (MDSI)

Property	TC05	TC06	RAS++
1. Filters (MT components)	Yes	Yes	Yes
2. Ports (Parameters in RAS++)	Yes	No	Yes
3. Artifact Ontology	No	Yes	No
4. MDE Settings	No	No	Yes
5. Any Type of Artifact	No	Yes	Yes
6. Artifact Federation	No	Yes	Yes
7. Light-weight Extensibility	No	No	Yes
8. Scenario 1	No	No	Yes
9. Scenario 2	No	Yes	Yes
10. Scenario 3 - OO	No	No	Yes
11. Scenario 3 - RDP	No	No	Yes

fies MDE Settings aiming at supporting the integration of MDE Artifacts through ports (BIEHL *et al.*, 2014) and filters (ZHANG, 2015). In some proposals for MDSI, MDE Settings are also embedded with concepts for Software as Services (SaaS) (BASCIANI *et al.*, 2014a), which allow the automatic access to MDE Artifacts on the cloud. These proposals are complementary to MTI because they consider techniques for integration of tools, while DSLs in the first classification only consider the catalog of MDE Artifacts. The scenario for MDE adoption is heterogeneous and also requires the use of several tools not based on models. An effective solution is to integrate these tools in MDE Settings (tool chain (AHO *et al.*, 2009)) through a DSL for tool adapters. Related with these concepts, instantiation of tools through Software as Service (SaaS) (BIEHL *et al.*, 2011, 2014; ROCCO *et al.*, 2015; WEIQING *et al.*, 2012; ZHANG and MOLLER-PEDERSEN, 2013; ZHANG *et al.*, 2012) gained attention in recent years. Aforementioned works propose the transformation from MDE Settings to specifications based on the Open Services for Lifecycle Collaboration (OSLC) (ELAASAR and NEAL, 2013), thus enabling the representation of MDE Settings in the cloud and instantiation through service connectors.

Table 5.22 compares three approaches for MDSI. In this sense, RAS++ use the term "parameter" instead of "port" to represent IO and "transformation component" of "component" instead of "filter" to represent some data transformation. In comparison with MDSI representations, such as TIL (BIEHL *et al.*, 2014), RAS++ is more expressive. On the other hand, authors in (ZHANG and MOLLER-PEDERSEN, 2014) argue that artifact ontology is fundamental for classifying MDE resources. We credited this need for a repository/service provider, thus not included in RAS++.

Artifact Typing

Artifact Typing in MDE (MDE-AT) classifies approaches that allow the representation of artifacts in a technical-level (VIGNAGA *et al.*, 2013) for consistency check. The intent of artifact typing is to provide data to handle valid compositions through parameter matching (YIE *et al.*, 2012), as well as check validity in the co-evolution

Table 5.23: Property table 6: Artifact Typing in MDE (MDE-AT)

LEGEND [Detailed=D], [Flexible=F], [S=Superficial]

Property	TC07	TC08	RAS++
1. Megamodel	Yes	No	Yes
2. MetaMetamodel	Yes	Yes	Yes
3. Metamodel	Yes	Yes	Yes
4. Ad-hoc MetaDefiniton	No	No	Yes
5. Model	Yes	Yes	Yes
6. Component	No	No	Yes
7. Transformation Component	Yes	Yes	Yes
8. MDE Settings	S	S	D
9. Toolbox	Yes	Yes	Yes
10. Abstraction for Serialization	No	No	Yes
11. Light-weight Extensibility	No	No	Yes
12. Scenario 1	No	No	Yes
13. Scenario 2	No	No	Yes
14. Scenario 3 - OO	No	No	Yes
15. Scenario 3 - RDP	No	No	Yes

of properties such as models, metamodels and transformations (VIGNAGA *et al.*, 2013).

Approaches for artifact typing, as illustrated in Table 5.23, have a similar goal to asset specifications. Existing proposals for MDE-AT are less rich in terms of representation of MDE Settings than the extensions that we have presented in RAS++. Moreover, RAS++ is based on concepts of assets, which means that reuse structures discussed in Table 5.17 increment existing approaches for MDE-AT for the globalization of information.

Model Transformation Chain

Model Transformation Chain (MTC) is the most common classification and allows the representation of semantics for multi-lifecycle. This classification includes DSLs that represent properties for model transformations and metamodels in a verifiable execution order (BASCIANI *et al.*, 2014b; YIE *et al.*, 2012) and model weaving (JOUAULT *et al.*, 2010).

The main goal of approaches for MTC is to check consistency in substitutions made in a representation of model-based operations. Table 5.24 compares RAS++ with two recent proposals for MTC, which propose the implementation of reuse mechanisms, out of the scope of a pivotal representation language. Thus, in terms of representativeness, RAS++ is well mapped for concepts needed in DSLs for MTC, in addition to representing a bigger diversity of data for scenarios 1, 2 and 3.

Component Models for MDE

Approaches for **Component Model for MDE (CMMDE)** intent to represent tool chains through reusable components, including properties for components, subcomponents, connectors (BASSO *et al.*, 2014e; CUADRADO *et al.*, 2014) and also properties for software product lines implementations (ARANEGA *et al.*, 2012a). Toolboxes in support for CMMDE allow the adaptation of components with concepts

Table 5.24: Property table 7: Model Transformation Chain (MTC)

Property	TC09	TC10	RAS++
1. Parameter Matching (Execution)	Yes	Yes	No
2. Substitutability (Execution)	Yes	No	No
3. Parameter Matching (Representation)	Yes	Yes	Yes
4. Substitutability (Representation)	Yes	No	Yes
5. Matching/Repository Rules	No	Yes	No
6. Heterogeneous Parameter Typing	No	No	Yes
7. Light-weight Extensibility	No	Yes	Yes
8. Scenario 1	No	No	Yes
9. Scenario 2	No	No	Yes
10. Scenario 3 - OO	No	No	Yes
11. Scenario 3 - RDP	No	No	Yes

Table 5.25: Property table 8: Component Model for MDE (CMMDE)

Property	TC11	TC12	RAS++
1. Runtime/Dynamic Adaptation	Yes	Yes	Yes
2. Fragmentation & Composition	Yes	No	Yes
3. Feature Model	Yes	Yes	No
4. Sub-typing	No	Yes	No
5. Instruction for Reuse	No	No	Yes
6. Light-weight Extensibility	Yes	No	Yes
7. Scenario 1	No	No	Yes
8. Scenario 2	Yes	Yes	Yes
8. Scenario 3 - OO	Yes	Yes	Yes
9. Scenario 3 - RDP	No	No	Yes

for software factories.

Table 5.25 shows FOMDA DSL (TC11) and Bentõ DSL (TC12) in comparison to RAS++. The first two lines show two reuse mechanisms implemented by these DSLs not supported in our work. In terms of representation, Bentõ DSL is limited to the execution of Feature Relationship of type XOR, while in the current format of RAS++ the Feature Model cannot be represented. Instead, our DSL has a more generic purpose in reuse, including instruction based on Variability Points and Usage Tasks through reusable assets. Considering that the bindings are reuse mechanisms out of the scope of a pivotal language, sub-typing that is used in Bentõ DSL is not included in RAS++. If needed, sub-typing can be represented as an individual artifact using extensibility mechanisms in RAS++. Thus, in our analysis, the absence of sub-typing in RAS++ does not compromise the aspect of representation of MDE Artifacts & Settings.

Scenario 5-Pivotal Representations

Except for the definition, few related representations are found in toolboxes for **Higher-Order Transformation (HOT)** shown in Table 5.26. This definition includes two types of works: for execution (GORP *et al.*, 2009) and for adaptation/reuse of model transformations. This way, we considered only the represen-

tational elements from those proposing binding specifications in orchestrations of model-based operations in higher-level than ATL, such as VIATRA2 (HORVÁTH *et al.*, 2006) and others (BASCIANI *et al.*, 2014a; CUADRADO *et al.*, 2014). Similarly as in MTCs, HOT includes elements usually specified in textual DSLs for chaining of model-based operations.

Table 5.26: Contributions with representations used for scenarios 5 and 6

Study	Paper	Title	Year
TC13	(HORVÁTH *et al.*, 2006)	Automatic generation of platform-specific transformation	2006
TC14	(BASCIANI *et al.*, 2014b)	Automated Chaining of Model Transformations with Incompatible Metamodels	2014
TC15	(POLGÁR *et al.*, 2009)	Model-based Integration, Execution and Certification of Development Tool-chains	2009
TC16	(MACIEL *et al.*, 2013)	Supporting model-driven development using a process-centered software engineering environment	2013
TC17	(LUCAS *et al.*, 2017)	CollabRDL: A language to coordinate collaborative reuse	2017

Table 5.27 shows RAS++ in comparison to pivotal representation approaches built on HOT concepts. As differential, HOT approaches are used for automatic generation of model transformations based on meta-data (metamodel). This reuse mechanism illustrated in Line 1 is very interesting, but it is out of the scope of RAS++, thus a limitation. Our justification is that HOT is a research topic more related to tool support for model transformation languages and barely related to the conceptual role of a pivotal language for MDE Artifacts & Settings.

Scenario 6-Software Process Integration

Although representations for process engineering are not included in this thesis, neither in RAS++, this scenario for tool chain is used for comparison. Properties here discussed align next representations to be introduced by our research group in RAS++. Table 5.28 compares RAS++ with approaches for integration of MDE Artifacts into Process Modeling Languages (PMLs) or Software Development Process (SDP): MDE-SDP. Line 1 shows a first difference: RAS++ is for representation, not for execution. Existing approaches, including (VARA *et al.*, 2014), allows the trace of artifacts along execution. Our DSL has a more general purpose than to represent SDPs. Anyway, in (BASSO *et al.*, 2014b) and `prisma.cos.ufrj.br/wct/tr07.pdf`, we introduced in RAS++ workflow elements as an ongoing work. Thus, in terms of representativeness, RAS++ can be used to represent MDE-SDP as well.

5.3.4 Conclusions

Based on our previous characterization studies for tool chain contributions, this chapter puts them side-by-side to evaluate "representativeness". In this sense, we

Table 5.27: Property table 9: Pivotal Representations

Property	TC13	TC14	RAS++
1. Conceptual Representation of MTs	Yes	Yes	No
2. Semantics & Syntax for MTC	Yes	Yes	Yes
3. General MDE Artifacts & Settings	No	No	Yes
4. Light-weight Extensibility	Yes	Yes	Yes
5. Scenario 1	No	No	Yes
8. Scenario 2	No	No	Yes
7. Scenario 3 - OO	No	No	Yes
8. Scenario 3 - RDP	No	No	Yes

Table 5.28: Property table 10: MDE & Software Dev. Processes (MDE-SDP)

Property	TC15	TC16	TC17	RAS++
1. General artifacts	Yes	Yes	Yes	Yes
2. Workflow elements	Yes	Yes	Yes	Yes
3. MDE Artifact Typing	Yes	Yes	No	Yes
4. Instruction for Reuse	No	No	Yes	Yes
5. Pattern Classification	No	No	No	Yes
6. Light-weight Extensibility	Yes	Yes	No	Yes
7. Reuse actions	No	No	Yes	No
8. Scenario 1	No	No	No	Yes
9. Scenario 2	No	No	No	Yes
10. Scenario 3 - OO	No	No	Yes	Yes
11. Scenario 3 - RDP	No	No	No	Yes

presented six scenarios for coopetition in MDE as a Service, arranging such contributions in specific intents for tool chain. Each scenario presents toolboxes allowing integration of MDE Artifacts in some contexts, where a context can be a target representation language, or a development environment, or a repository/database model. Then, inside each scenario, a property table is shown and compares at least two toolboxes competing for adoption. Into the tool chain context and outside the boundaries of each scenario, toolboxes can cooperate. Thus, some toolboxes are complementary to help Software Engineers in integration processes for MDE.

So far, integration is considered feasible by researchers/developers of such toolboxes, but always considering the homogeneity for representation of data from their own set of tool support. There is no support to integrate them all in a coopetition scenario. Since previous chapter demonstrated that manual integration is costly, automated integration through a pivot is an interesting solution. This requires the selection of a pivot language to be used by converters.

In this sense, a wrong choice can imply in a undesired side-effect: loss of data when applying automatic integration mechanisms. Representations from the stat-of-art, presented here as a set of toolboxes and properties, are not representative enough to be selected as pivot for conversions. This means that they can work well for part, but not all, the 10 demonstrated intents in tool chain for MDE. In this sense, RAS++ is more adequate since it is more representative than the other

options.

KRIPPENDORFF (2010) goes further and claims that *"The disadvantage of a pivot language is that each step of re-translation introduces possible mistakes and ambiguities."*. Mistakes and ambiguities must be avoided in future implementations of coopetition in MDE as a Service. A feasible solution is validate asset constructs, as illustrates Figure 5.26, one of the advantages in building a pivot language as a DSL generated with EMF (STEINBERG *et al.*, 2008).

In this sense, we found that the following elements, discussed in details in the *Transformation* phase, contribute for reducing mistakes and ambiguities inherent from conversions built on a pivot language: 1) Well-formedness rules, expressed in OCL constraints, allows validating objects built on RAS++ metaclasses; 2) The grouping of common, but ambiguous properties (e.g., Filters versus Transformation Components, Ports versus Transformation Parameters, Parameter Types and Artifact Types, and others), and; 3) Specific metaclasses for MDE Artifacts and Settings with constructors found in real settings for tool chain. These are benefits not found in asset-related contributions, thus points in favor to RAS++ in future implementations of coopetition in MDE as a Service.

Finally, we conclude that RAS++ contains properties considered important by many tool chain contributions. For this reason, it is more representative than toolboxes for integration. Thus, Software Engineers will minimize loss of data derived from the execution of automatic conversions when selecting RAS++ as a pivot.

5.4 Thought Experiment

This section presents final remarks considering an analysis for approaches that also could promote coopetition in MDE as a Service. For instance, this scenario is too complex for evaluation through case studies, which would exceed, by a large amount, the time available in a unique PhD. Thus, our best alternative to compare alternatives is through a thought experiment (YEATES, 2004).

This study is organized as follows. Section 5.4.1 presents the study goal and Section 5.4.2 the adopted research method. Section 5.4.3 depicts the analysis and Section 5.4.4 draws conclusions.

5.4.1 Goal

Our goal is to reach some reasons why RAS++ should be considered in future implementations of coopetition, as describes Table 5.29. Our general research question is **Q9: Why adopt RAS++ in future implementations of MDE as a Service?** Likewise, we search answers to the following specific research questions:

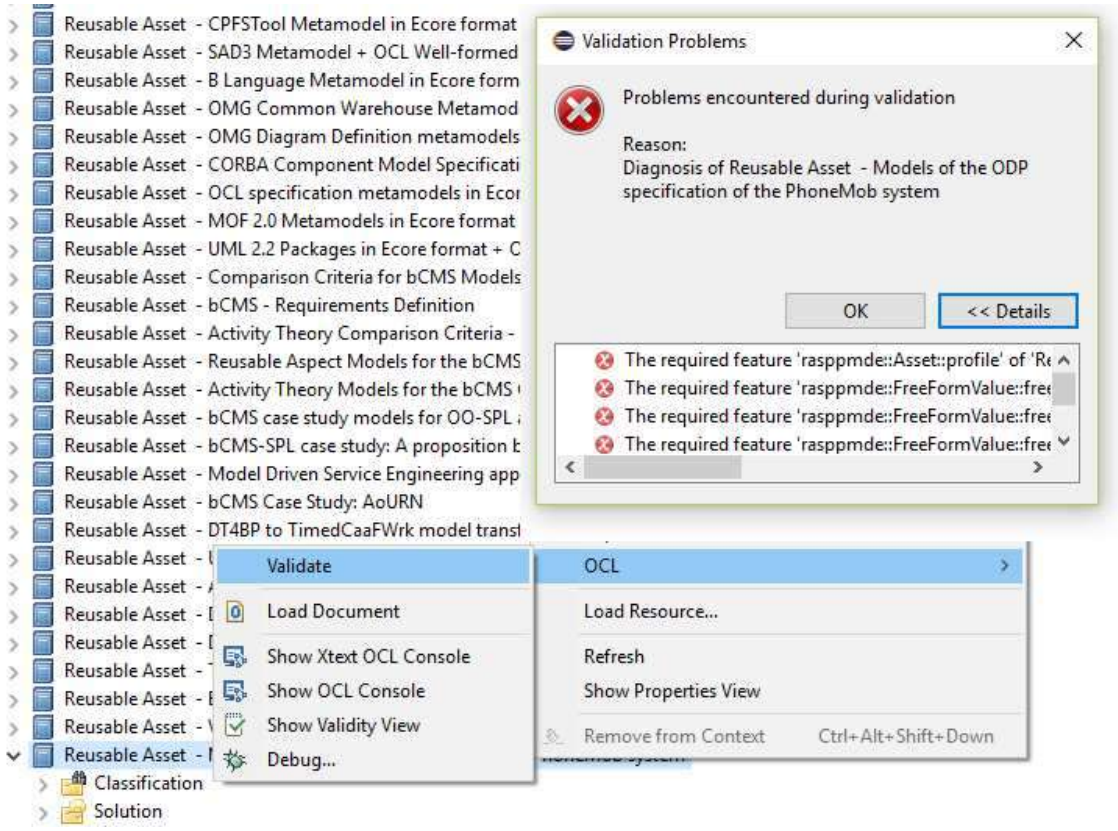


Figure 5.26: Validation of asset representations.

- RQ1 - Why not simplify the scenario by using a unique MDE repository?
- RQ2 - Are we able to, by using the simplest scenario as possible, automatically integrate assets for MDE into a tool chain?
- RQ3 - Is the use of model-to-model transformations, applied directly between different MDE Settings, the best way for automatic integration?
- RQ4 - Why manual integration is not an interesting solution for implementation of coopetition in MDE as a Service?
- RQ5 - Why haven't you elected a pivotal language from the state of the art?
- RQ6 - Without a common representation language, what is the consequence for future implementations of coopetition in MDE as a Service?

Table 5.29: Thought experiment goal of the motivated scenario

Analyze	The scenario for coopetition in MDE as a Service
For the purpose of	Characterizing
Regarding	Challenges for implementation
From the viewpoint of	Researcher
In the context of	Development costs for connectors.

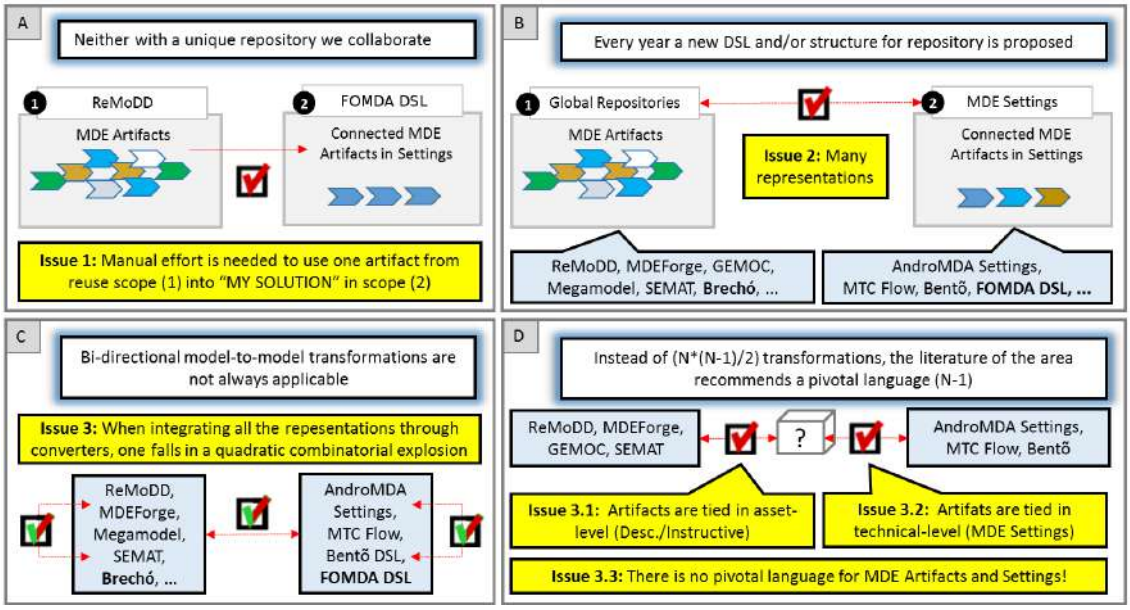


Figure 5.27: Issues for integration found in hybrid assets.

5.4.2 Research Method

We applied a thought experiment (BROWN and FEHIGE, 2017). According to YEATES (2004), a thought experiment is the way to perform a structured process of intellectual deliberation, speculating about potential consequences or antecedents of some specifiable problem domain. Thus, we present our findings built upon previous experiences, assessments and mappings from the literature by exercising what is also expected from a PhD student: the logical thinking.

5.4.3 Analysis

In order to answer these questions, Figure 5.27 shows four different scenarios that we have been considering for execution of though experiment. These scenarios range a simple integration issue to more complex ones, thus reflecting the current challenges for implementation of coopetition in MDE as a Service.

RQ1 - Why not simplify the scenario by using a unique MDE repository? Because a unique repository/platform **will not reduce the difficulty for implementation.** The usage of a unique repository (or MDE platform) is suggested as a possible a way to obtain collaboration in MDE (ROCCO *et al.*, 2015). Since ReMoDD (FRANCE *et al.*, 2007) was proposed with such intent too, and based on the ReMoDD experience after the data mining discussed in previous sections, we conclude that the state of the practice will not make a big effort to promote a unique infrastructure in support for assets. For example, considering the simplest scenario for coopetition illustrated in Figure 5.27 (A), which considers ReMoDD as the unique platform for assets, an implementation for MDE as a Service could

take place right now. Besides, differently from Amazon that is a leader in terms of platform for bookstore (RITALA *et al.*, 2014), we have no leadership in terms of MDE repository that plays the role of a platform where coopetition takes place like Amazon web services. Thus, the practice shows us that simplify the scenario by electing a unique repository is not the solution neither is this under consideration by research community.

RQ2 - Are we able to, by using the simplest scenario as possible, automatically integrate assets for MDE into a tool chain? Considering the simplest scenario for our implementation, assume that FOMDA DSL is used for integration of tool chains. For us, this is the ideal scenario since we are experts in this language and associated tool support. In this scenario, Software Engineers analyze Artifacts shared in ReMoDD, as illustrated in Figure 5.27 (A.1), acquire and introduce some in tool chains built on a unique format (FOMDA DSL), as illustrated in Figure 5.27 (A.2). This sequence of actions seems to be easy to implement, as we have first proposed in (BASSO *et al.*, 2013c). However, we have found that even in this simplest scenario, it is still not possible to apply automatic integration of MDE Artifacts into the FOMDA DSL, since artifacts in ReMoDD are hybrid. Thus, due to the lack of a common representation in ReMoDD for technicalities needed in FOMDA DSL, this implies in manual effort along executions of three preliminary phases for tool chain.

ReMoDD does not represent technicalities needed in FOMDA DSL, but we found that MDEForge (ROCCO *et al.*, 2016) allows the representation of some. For this reason, MDEForge is best tailored for MDE Artifacts and Settings than ReMoDD. In this case, a partial integration is possible, but not without loosing information. An alternative would be electing a more representative language as pivot for transformations (CUADRADO *et al.*, 2014), resulting in RQ3.

RQ3 - Is the use of model-to-model transformations, applied directly between different MDE Settings, the best way for automatic integration? The answer is no, because **the real scenario is too much heterogeneous, implying in loss of information too.**

The scenario shown in Figure 5.27 (B) reflects the reality that we found in terms of heterogeneity. In (1) many options for repositories (e.g., SEMAT, ReMoDD, GEMOC, and others, such as Eclipse MDT and the embryonic MDEForge) are available. In (2) many representations for MDE Settings are also available. In this regard, it is needed to consider that: a) MDE Artifacts are always tied to specific representations from reuse and execution mechanisms (1 or 2), and b) properties from representation languages are not the same.

In an hybrid reuse scenario like this, implemented without a common representation language, an automatic integration would demand manual conversions from all

the possible representations for artifacts. Therefore, a second issue for coopetition in implementation of MDE as a Service is that, currently, too many hybrid representations are available, making direct conversions less feasible as more and more representations are included in the problem domain.

RQ4 - Why manual integration is not an interesting solution for implementation of coopetition in MDE as a Service? If the usage of direct conversions is somehow not feasible, manual transformations are even worst for integration. So, the simple answer is: **due to reuse costs involved, manual integration in this scenario is not the ideal.** In this sense, a manual effort for conversion between representations for MDE Artifacts and their Settings will imply in extra reuse costs (BECKER *et al.*, 2007). A possible solution to reduce reuse costs is by implementing a reuse mechanism that connects this hybrid reuse scenario with an homogeneous local reuse scenario. This is not possible because, currently, the reuse opportunities are distributed in many repositories. Another solution would be automatic conversion between different representations for MDE Artifacts and Settings, presented before, which can be implemented though model-to-model transformations (BÉZIVIN, 2005), but too many transformations are needed to integrate all possible representations. This issue tends to increase in complexity when a new representation is added into the coopetition scenario.

RQ5 - Why haven't you elected a pivotal language from the state of the art? In previous experiences (BASSO *et al.*, 2014e), we found that **electing a pivotal language does not reduce the complexity for integration because languages in the state of the art are not representative enough.** For example, Figure 5.27 (C) illustrates the last implementation alternative. In this scenario, coopetition is assisted through the execution of bi-directional model-to-model transformations by electing a pivot (one of the options). In this case, it is needed the development of connectors for all the possible representations with the selected pivot. As discussed earlier, this is a better solution than a manual integration approach. However, the lack of an expressive representation language in the state of the art implies in loss of information across conversions, thus a undesirable consequence for automatic integration.

Besides, even when applied with model transformation reuse concepts (KUSEL *et al.*, 2015), model-to-model transformations do not cover the handcrafted representations for MDE Artifacts & Settings. For example, in a previous case study (BASSO *et al.*, 2014e), we observed that bi-directional transformations with modern model transformation engines are not always applicable, such in conversions from FOMDA DSL to settings in AndroMDA (MONTEIRO *et al.*, 2014a) that are represented in XML[1]. In this case, for each domain model represented in FOMDA DSL, we devel-

[1]AndroMDA web page <http://andromda.sourceforge.net/>

oped four model transformations to integrate MDE Artifacts in settings distributed in four XML representations. Thus, the real scenario presents too many possible combinations, exploding the number of model transformations to a bigger number than found in a quadratic combinatorial explosion.

RQ6 - Without a common representation language, what is the consequence for future implementations of coopetition in MDE as a Service? The consequence is an increasing difficulty for implementation of coopetition in MDE as a Service, since **we are falling in a quadratic combinatorial explosion issue.** This means that even recent proposals for model transformation reuse, such as Bentõ DSL (CUADRADO *et al.*, 2014) that is focused in generating model-to-model transformations from higher-level concepts, has a delimited scope for conversion and does not cover the reuse of settings tied to artifacts. Meanwhile, many other settings are reported to be represented in database structures (BASCIANI *et al.*, 2014b), textual files (HORVÁTH *et al.*, 2006), code (VÖELTER and GROHER, 2007), Excel Spreadsheets (RESCHENHOFER and MATTHES, 2015). These are relevant representations associated with MDE Artifacts that, in the context of a future MDE platform (COMBEMALE *et al.*, 2014), need a common format for reuse in global scale.

Hence, the absence of a common format would imply on the development of many converters for integrating architectural representations of these settings. For example, in a worst case scenario (i.e., using ad-hoc representations for settings), the implementation of MDE as a Service would imply in combinatorial explosion for conversions: i.e., needing the development of at least `(N*(N-1))/2` converters, where N is the number of representation formats.

5.4.4 Conclusions

It is difficult to reach common representations in a big scenario without incurring in basics. For example, in order to be non restricted to specific MDE Artifacts, ReMoDD adopted only classifications to represent assets (FRANCE *et al.*, 2007). As an undesired result, ReMoDD's assets are impossible to be used by conversions to allow the automatic integration of asset content into tool chain representations. An alternative that could help to reduce the number of combinations is the advent of a more representative and common Knowledge Base (KB) (MUSSBACHER *et al.*, 2014). A KB has been pursued by the MDE community since 2007 with the ReMoDD and, more recently, by the initiate GEMOC (COMBEMALE *et al.*, 2014). SEMAT (JOHNSON *et al.*, 2012) is another example, an initiative that is creating a KB in Software Engineering using the Essence (OMG, 2015) as a core/pivotal representation language. SEMAT is limited for specifications of methods, it is out of

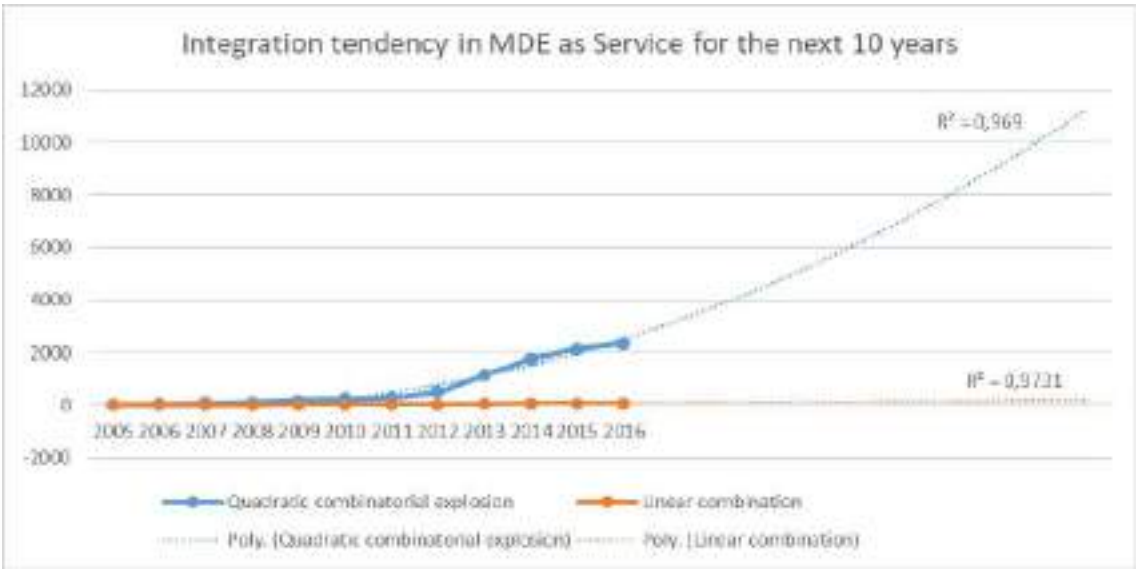


Figure 5.28: Integration effort tendency in tool chain for the next 10 years.

the scope in this thesis and it cannot be used to represent data from MDE Artifacts and Settings. GEMOC, on the other hand, is mirroring on the SEMAT, but keeping focus on the MDE Artifacts, which is more directed for collaboration in the MDE as a Service. Although we considered GEMOC proposal important to centralize assets for MDE better than currently is observed in ReMoDD, GEMOC is also limited to play the role of a pivotal language in a coopetition scenario for MDE as a Service.

In this point of our analysis, we make a parallel to the study presented in previous chapter. Figure 5.28 shows a tendency for development of connectors with a perspective for the next 10 years. Our conclusion is that, so far and except for RAS++, there is no adequate language for pivoting the 69 representations found in our literature mappings. Without an appropriate language, one could fall in a quadratic combinatorial issue, where these 69 representations would imply in 2.346 combinations to enable coopetition today. In 10 years from now, this effort increases to more than 11.000 combinations. This is a considerable effort for development that hampers initiatives currently and certainly will hamper in the future. We found that this number of combinations is not currently feasible. Therefore, the necessary effort for development of connectors without a pivotal language would scare away cooperation initiatives.

Another parallel is to our characterization studies (BASSO *et al.*, 2015b, 2016d, 2017a). Considering that the development of model transformation is costly (HUTCHINSON *et al.*, 2011), one should avoid this issue whenever possible. For example, along four years (BASSO *et al.*, 2014a) Adapit team developed less than 400 model transformations in a local implementation for MDE as a Service, i.e., without any integration issue derived from coopetition scenarios. Based on the

analyzed tendency, coopetition is out of consideration for implementation. On the other hand, an integration supported by 68 connectors allowed by RAS++ is encouraging, as well as the tendency for a linear combination observed in Figure 5.28, which is stable as new representations are introduced into the integration issue.

So ... **Why adopt RAS++?** One should adopt RAS++ because it reduces the problem to a linear combinatorial problem, thus more manageable and feasible in terms of integration for tool chain. As illustrates Figure 5.27 (C), connectors are essential for coopetition, presenting a lower cost for integration than in manual effort for representation of tool chains (BASSO *et al.*, 2014e; CUADRADO *et al.*, 2014). However, due to the heterogeneity of scenario, a direct conversion between candidate representations is not the ideal. Instead, a common representation language (RAS++) simplifies this implementation, as illustrates Figure 5.27 (D).

A final consideration is that, for other representations than tool chain, Software Engineers would need to evolve RAS++. Our contribution is limited and similar endeavors should consider other scenarios and representations.

Chapter 6

Conclusions

> I believe in intuitions and inspirations...I sometimes FEEL that I am right. I do not KNOW that I am.
>
> Albert Einstein

6.1 Summary

MDE Artifacts are produced for diverse intents, such as reuse in intra and inter-organizational contexts. Some MDE Artifacts are shared globally with intents to share knowledge or to support new business opportunities through cooperation and competition (coopetition). In order to be used in software projects or processes, these artifacts are typically configured in chain of tool, or tool chains. Despite a large availability of tool support, the literature agrees that it is always very difficult to integrate these artifacts into a tool chain. Recent evidence suggests the need for preliminary work/phases on tool chain that, in addition to technical information associated with MDE Artifacts, should consider descriptive information. However, there is no material in the literature reporting on what information should be represented. Thus, requirements for execution of these preliminary phases are not properly introduced by the current literature, thus characterizing an open research topic for investigation.

In this sense, research on the MDE adoption has reported that several challenges still hinder the delivery of a tool chain. It is acknowledged that, in a building process for tool chain, Software Engineers perform tasks such as classifying, selecting, adapting, and introducing MDE Artifacts through chains that fit to the requests from target contexts. In this sense, target contexts can be software projects, teams, companies, technologies, and others. Since a tool chain is considered as a final

product in this thesis, our contributions are for supporting earlier phases. Therefore, we characterized three preliminary phases to the definition of the tool chain: Specification, Acquisition and Transformation.

As our main contribution, we built a common representation language for the execution of these phases: the RAS++. This language is result of a mapping of metaclasses and properties found in many representations adopted in scenarios similar to MDE as a Service. To the Specification phase, we considered properties from two asset specification languages: RAS and AMS, thus mapping requirements for representation of MDE Artifacts in modern repositories. To the Acquisition phase, we have considered data for decision making from different sources, thus mapping properties that bring coopetiton in MDE as a Service including classifications for MDE Artifacts and in representations adopted in SECO approaches, with benefits and drawbacks discussed in Section 6.2.2. The last phase is called Transformation and allows the representation of technical data for tool chain in assets, with an analysis of representativeness discussed in Section 6.2.4.

Through these phases, we have successfully tested RAS++ in real settings: two scenarios by mining homemade assets and one global reuse scenario by mining ReMoDD repository. With this, we conclude that RAS++ is representative and relevant for one MDE repository proposed by experts in the area. Besides, to reach relevant metaclasses and properties in RAS++, an extensive research mapping was carried out. To the Transformation phase, for example, we had to consider diverse tool chain representations adopting hybrid reuse mechanisms. We found that MDE Artifacts are connected through representations/DSLs characterized by MDE Settings. In some sort of tool support, reuse mechanisms for tool chain allow you to specify hybrid properties such as chains of transformations and their components, processes and their tasks, ports and interfaces of integration, as well as variability associated with models, metamodels and transformations. Therefore, RAS++ was built over data extracted from reliable data sources.

By using MDE Settings, a reuse or integration mechanism implements operations for introducing artifacts in target contexts. Reaching common representations in this matter required a big effort for mapping of common technical properties and intents from tool support. For example, such operations also depend on the intentions of those mechanisms, which means that they can cooperate in a tool chain instantiation process. Some proposals for tool support aim at assisting the specification and adaptation of software development process models, others in the reuse of model transformations, some in the consistency check in the integration of more than one DSL in a chain of transformations, among other intentions. These tools can be divided in two field of studies: system engineering and process engineering. Due to the focus of this thesis, the last field of research is not considered in our mapping.

We also found that, currently, existing tool chain initiatives neglect preliminary approaches in the literature of the area. We found only recently research positions claiming the importance of analysis of descriptive information in addition to the technical information adopted in tool chain reuse mechanisms. Technical data associated with the artifacts and their configurations are also critical to succeed, thus needing a link between technical and descriptive data. This link is missed in the literature, which could be supported by a common representation. However, asset representations are too much general and MDE Setting DSLs are too limited to be used as a common representation language. Thus, we proposed a connection between these two ways to represent data associated with MDE Artifacts, with benefits discussed in Section 6.2.2.

Our proposal simplifies the execution of preliminary phases for tool chain. Though preliminary studies, we found that the diversity of representations for technicalities and description of MDE Artifacts hamper the use of automatic techniques for tool chain (i.e., the use of model transformations). This is because too many transformation are needed to integrate all the investigated scenario. This may be a reasonable explanation why Software Engineers take manual approaches to integration. As a contribution to this issue, a pivotal or common language like RAS++ allows the use of automatic techniques through model transformations. RAS++ can represent data from various MDE Artifacts found in the repositories and the various representations for MDE Settings used in reuse mechanisms for the tool chain. Therefore, RAS++ is characterized as a pivotal language, reducing a quadratic combinatorial issue for integration into a linear issue, as discussed in Section 6.2.2.

We introduced concepts in reusable assets that are mapped to many proposals for MDE Artifacts and Settings found in the literature. This makes our contribution different from all the other known representation languages proposed in software engineering. We conclude that RAS++ is representative for the hybrid assets, as motivated in this thesis. Existing languages are less representative for our long term goal in MDE as a Service: implement coopetition. Thus, the operational definition for a common representation language reflects the true meaning of the theoretical concept from hybrid representations here discussed.

6.2 Discussions

6.2.1 Contributions

Our main contributions are:

- **Evidences suggesting the need of a common/pivotal representation language**. These evidences are exclusive contributions from this thesis, thus

not found in the current literature.

- **A new DSL called RAS++**. This language promotes some benefits not allowed by other DSLs and proposals discussed in the Related Works section.

- **Characterization of MDE as a Service.** Our publications discussing MDE s Service allows: 1) A better understanding of issues hampering the introduction of MDE in contexts of start-up companies; 2) A better understanding of reuse mechanisms that gave good results in implementation initiatives, and; 3) Benefits and drawbacks found in practice.

Our secondary contributions include:

- **Experience reports.** This research is constructed based on an insight discussed in three software reuse disciplines, in 2012, uttered by prof. Cláudia Werner. In the occasions, some issues for introduction of MDE in software development companies have been reported. In the absence of reports in the literature, a contribution derived from this thesis is a set of experience reports for challenges in conducting real MDE-based software processes.

- **Mined data.** In order to evidence these issues, it was mined different elements from five real MDE-based software projects developed in three companies that used a specific tool support for implementation of MDE as a Service. Besides, it was mined data from the ReMoDD repository. These studies not only proved that MDE as a Service is a real scenario as provided data for future analysis.

- **Development of new tools in support for RAS++:** 1) A prototype to deploy the content of assets in DSLs to MDE Settings and to Eclipse IDE; 2) A prototype for automatic retrieval of metadata in reusable assets, and; 3) A prototype with features for publishing and downloading assets to/from a global repository conforms to RAS.

- **Small advance in integration of RAS++ with an integration language from our research group (RDL).** Properties introduced in RAS++ in support for the Transformation phase are not restricted for tool chain scenarios. In this sense, we started some tests for use of RAS++ as a common representation language in representations of object oriented assets. So far, it is possible to integrate RAS++ with RDL (OLIVEIRA *et al.*, 2011), allowing RDL an independence of representation languages for OO programs (UML, Ecore, Source Code, JDT, and others). Thus, besides tool chain, we tested RAS++ as a pivotal language on the top of reuse processes for Object Oriented Framework Instantiation (OOFI) (LUCAS *et al.*, 2017).

- **Improvements in previous tool support.** It was possible to improve three tools developed before this PhD: WCT, FOMDA DSL and MockupToME.

6.2.2 Benefits

RAS++ can foment new initiatives through some benefits associated with a pivotal representation. This language must connect two reuse scopes: 1) abstractions of MDE Artifacts and also their physical content are stored in database structures. These structures are representations found in the literature as metamodels, thus belonging to a set of languages "A"; 2) the state of the practice that presents several DSLs for MDE Settings, generalized as a set of languages "B". These languages allow to introduce integrated artifacts in target context for adoption in, what we defined as, a first reuse scope of MDE artifacts.

In a third reuse scope, RAS++ complements the state of the art by connecting MDE Artifacts and with MDE Settings with the following benefits:

Unification of Representational Concepts

Representational concepts from consumers and providers of MDE Artifacts and Settings. The state of the art presents contributions that are associated with repositories or with execution of model-based operations. Abstracting these representations for a level of assets allows to manage this scenario with data used by consumers and providers of artifacts. Besides, bringing assets into a design issue instead of database structures makes the development of tool support easier. This is because we can now use several other tool support in MDE, managing representational issues in models. Thus, we cannot impose limits for contributions that will come up from this work.

Coopetition through a common language. The execution of any coopetition scenario requires a common language shared by companies/professionals (RITALA *et al.*, 2014). There is no common language for coopetition in MDE as a Service. For example, the representation of MDE Artifacts is usually associated with a setting, which are limited to consider elements from assets discussed for Specification and Acquisition. This makes the artifact dependent on a specific representation and, in the same way, hampers the usage of a MDE Setting DSL as a common language. Instead, a common representation language would allow a coopetition through common properties from different DSLs, thus scaling the usage of MDE Artifacts & Settings in at least three phases: Specification, Acquisition and Transformation.

Generality for structural features. Because these phases are classified in the MDE context, RAS++ should be capable of representing abstractions for MDE

from all these types of artifacts as well. This is demonstrated through asset representations used along this thesis.

Independence from a repository vendor. Another advantage in specifying information in assets is that it is independent from a repository vendor. In addition to RAS and AMS, RAS++ supports both representations of assets. Thus, one can change the service provider that hosts the artifacts without losing asset information.

Applicable to describe almost every type of MDE Artifact currently stored in repositories/clouds. In our analysis of possibilities for integration, we found that asset specifications can be used to describe any other component used in MDE platforms, reuse repositories, service providers, the SEMAT knowledge base, and others. One can consider for representation any technical solution (system) non strictly for MDE, or even techniques and didactic materials for Software Engineering in general. Some initiatives are using assets specified with AMS to describe tools provided on the cloud such as database management systems, application servers and custom applications. This is a benefit that can promote the implementation of coopetition in MDE as a Service in the future.

Reduction for a Linear Combinatorial Issue

The current theory claims that a pivotal language "P" can enable integration between different representations. For example, between "A" and "B", with "A" being a tool chain representation and "B" a transformation component representation. Thus, by means of "P=RAS++", the integration in a tool chain can occur by means of few automatic transformations, since the different representations are integrated in a common format "P".

Due to common concepts, RAS++ allows to convert representations from a set of languages "A" to a pivotal "P", and then from "P" to another set "B". According to the current computational theory[1], a pivotal "P" would avoid this issue with a linear combination (N-1), rather than quadratic `(N*(N-1))/2`. Likewise, instead of transforming directly from "A" to "B", as found in the current stage of practice, we propose reusers to develop model transformations from "A" and "B" to "P".

6.2.3 Limitations

Since the evaluation in this thesis focused on ReMoDD resources, it is not possible to conclude on the feasibility of RAS++ for software ecosystem platforms and other repositories. We believe that RAS++ is feasible regardless of the platform used. However, unless we test our proposal in such scenarios, feasibility remains a limitation.

[1] https://en.wikipedia.org/wiki/Pivot_language

Another limitation of this thesis is that we are assuming that the metaclasses for descriptor groups from the standard RAS are representative for search and comparison of assets in *Acquisition* phase. In a previous work, we claimed that a better structure for descriptive information could benefit reusers for comparison of assets (BASSO *et al.*, 2016a), which is important for decision making in a coopetition scenario for MDE as a Service. In new studies, we found properties for decision making to be used in Acquisition phase (BASSO *et al.*, 2017c), requiring now a new investigation to the following open question: are these descriptive properties better represented using existing classification mechanisms from RAS or using new structures for RAS++?

To answer this open question, an empirical evaluation with an appropriate focus group is needed. Due to this limitation in our study, representations for the *Acquisition* phase are not included in this thesis, thus opening window for improvement in future works.

We also started developing some integrations with repositories, but we suspended them to focus on theory rather than on an engineering solution. For example, focusing on conceptualization of a common representation language instead of development and evaluation of tool support and web services. Thus, future works should overcome this limitation by enhancing and evaluating tool support, integrating RAS++ with ReuseECOS repository (SANTOS and WERNER, 2012) (a software ecosystem platform).

6.2.4 Threats to Validity

A threat to validity relies in our literature reviews shown in Table 6.1, which are structured mappings. They should not be confused with systematic literature reviews neither with systematic mapping studies, which include more than one researcher in the overall process of analysis. Since a unique researcher decided which property should be inserted in RAS++, this is an important threat to validity for the entire contribution. Our rationale for executing structured mappings is that so far we have had difficulties in bringing together researchers at the round table. The difficulties are mainly due to the necessary technical level and the few schedules available for the application of systematic mappings. To remove this bias, we brought in three researchers into the follow-up of this research, now requiring a re-execution of our studies with a systematic mapping protocol.

Table 6.1: Mapping studies for RAS++ conception

Id	Title	Available at
M01	Characterizing the "MDE as Service" Research Agenda	prisma.cos.ufrj.br/wct/ms01.pdf
M02	Semantic Properties of Software Components	prisma.cos.ufrj.br/wct/ms02.pdf
M03	Intents from Asset Platforms and Their Properties	prisma.cos.ufrj.br/wct/ms03.pdf
M04	MDE Settings Intents and Their Properties	prisma.cos.ufrj.br/wct/ms04.pdf
M05	Diversity of MDE Toolboxes and Their Uncommon Properties	prisma.cos.ufrj.br/wct/ms05.pdf
M06	A Criteria for Representation of Technicalities from MDE Settings and Toolboxes	prisma.cos.ufrj.br/wct/ms06.pdf

Id	Review Type	Research Protocol	Reviewers	Sources
M01	Structured Mapping Study	Snowballing	1	Researchgate
M02	Ad-hoc Mapping Study	Key-wording	1	ACM, Researchgate
M03	Ad-hoc Mapping Study	Multi-vocal	1	Multiple voices
M04	Structured Mapping Study	Key-wording	1	Scopus, Researchgate
M05	Structured Mapping Study	Snowballing	1	Scopus, Researchgate
M06	Structured Mapping Study	Coding	1	Previous mappings

References

AGNER, L. T. W., SOARES, I. W., STADZISZ, P. C., et al., 2013, "A Brazilian Survey on UML and Model-driven Practices for Embedded Software Development", *J. Syst. Softw.*, v. 86, n. 4 (abr.). doi: 10.1016/j.jss.2012.11.023.

AHO, P., MÄKI, M., PAKKALA, D., et al., 2009, "MDA-based tool chain for web services development". In: *4th Workshop on Emerging Web Services Technology*, WEWST '09, pp. 11–18. ISBN: 978-1-60558-776-9. doi: 10.1145/1645406.1645409.

AIGNER, M., ZIEGLER, G. M., 1998, *Proofs from THE BOOK*. Berlin, Heidelberg, Springer-Verlag.

AKYILDIZ, I. F., SU, W., SANKARASUBRAMANIAM, Y., et al., 2002, "Wireless sensor networks: a survey", *Comput. Netw.*, v. 38 (March), pp. 393–422. doi: 10.1016/S1389-1286(01)00302-4.

ALEGRÍA, J. A. H., BASTARRICA, M. C., QUISPE, A., et al., 2011, "An MDE Approach to Software Process Tailoring". In: *2011 International Conference on Software and Systems Process*, ICSSP '11, pp. 43–52.

ALVAREZ, C., CASALLAS, R., 2013, "MTC Flow: A Tool to Design, Develop and Deploy Model Transformation Chains". In: *Workshop on ACadeMics Tooling with Eclipse*, ACME '13, pp. 7:1–7:9. ISBN: 978-1-4503-2036-8. doi: 10.1145/2491279.2491286.

ALVES, V., GHEYI, R., MASSONI, T., et al., 2006, "Refactoring Product Lines". In: *5th International Conference on Generative Programming and Component Engineering*, GPCE '06, pp. 201–210. doi: 10.1145/1173706.1173737.

AMBLER, S. W., 2015, *Approaches to Agile Model Driven Development (AMDD)*. Technical report, Agile Modeling. Available at: <`http://www.agilemodeling.com/essays/amddApproaches.htm`>.

AMS, 2014, 2014. "Asset Management Specification. Av. at <http://open-services.net/wiki/asset-management/OSLC-Asset-Management-2.0-Specification/>". .

ANDERSSON, P., HST, M., 2008, "UML and SystemC - A Comparison and Mapping Rules for Automatic Code Generation". In: *Embedded Systems Specification and Design Languages*, v. 10, pp. 199–209.

ARANEGA, V., ETIEN, A., MOSSER, S., 2012a, "Using Feature Model to Build Model Transformation Chains". In: *15th International Conference on Model Driven Engineering Languages and Systems*, MODELS'12, pp. 562–578, a.

ARANEGA, V., ETIEN, A., MOSSER, S., 2012b, "Using Feature Model to Build Model Transformation Chains". In: *Model Driven Engineering Languages and Systems*, v. 7590, *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 562–578, b. ISBN: 978-3-642-33665-2. doi: 10.1007/978-3-642-33666-9_36. Available at: <http://dx.doi.org/10.1007/978-3-642-33666-9_36>.

ASZTALOS, M., SYRIANI, E., WIMMER, M., et al., 2011, "Simplifying Model Transformation Chains by Rule Composition". In: Dingel, J., Solberg, A. (Eds.), *Models in Software Engineering*, v. 6627, *Lecture Notes in Computer Science*, pp. 293–307. Springer Berlin Heidelberg. ISBN: 978-3-642-21209-3. doi: 10.1007/978-3-642-21210-9_28. Available at: <http://dx.doi.org/10.1007/978-3-642-21210-9_28>.

AXELSSON, J., PAPATHEOCHAROUS, E., ANDERSSON, J., 2014, "Characteristics of software ecosystems for Federated Embedded Systems: A case study", *Information and Software Technology*, v. 56, n. 11, pp. 1457–1475. Special issue on Software Ecosystems.

AZANZA, M., BATORY, D., DÍAZ, O., et al., 2010, "Domain-Specific Composition of Model Deltas". In: *3rd International Conference on Model Transformations (ICMT 2010)*, pp. 16–30. ISBN: 978-3-642-13687-0.

BADAMPUDI, D., WOHLIN, C., PETERSEN, K., 2016, "Software component decision-making: In-house, OSS, {COTS} or outsourcing - A systematic literature review", *Journal of Systems and Software*, v. 121, pp. 105 – 124.

BASCIANI, F., ROCCO, J. D., RUSCIO, D. D., et al., 2014a, "MDEForge: An extensible Web-based modeling platform". In: *2nd International Workshop*

on Model-Driven Engineering on and for the Cloud, CloudMDE 2014, pp. 66–75, a.

BASCIANI, F., RUSCIO, D. D., IOVINO, L., et al., 2014b, "Automated Chaining of Model Transformations with Incompatible Metamodels". In: *Model-Driven Engineering Languages and Systems*, v. 8767, *Lecture Notes in Computer Science*, Springer International Publishing, pp. 602–618, b. ISBN: 978-3-319-11652-5. doi: 10.1007/978-3-319-11653-2_37. Available at: <http://dx.doi.org/10.1007/978-3-319-11653-2_37>.

BASSO, F. P., PILLAT, R. M., OLIVEIRA, T. C., et al., 2013a, "Supporting Large Scale Model Transformation Reuse". In: *12th International Conference on Generative Programming: Concepts & Experiences.*, GPCE'13, pp. 169–178, a.

BASSO, F. P., WERNER, C. M. L., PILLAT, R. M., et al., 2013b, "A Common Representation for Reuse Assistants". In: *13th International Conference on Software Reuse*, ICSR'13, pp. 283–288, b.

BASSO, F. P., WERNER, C. M. L., PILLAT, R. M., et al., 2013c, "How do You Execute Reuse Tasks Among Tools? A RAS Based Approach to Interoperate Reuse Assistants". In: *25th International Conference on Software Engineering and Knowledge Engineering*, pp. 721–726, c. Available at: <http://index.ksi.edu/conf/seke/2013/cr/243.pdf>.

BASSO, F. P., PILLAT, R. M., OLIVEIRA, T. C., et al., 2014a, "Newsletter - Supporting Large Scale Model Transformation Reuse", *ACM SIGPLAN Notices*, v. 49, n. 3, pp. 169–178. doi: 10.1145/2637365.2517218.

BASSO, F. P., WERNER, C. M. L., OLIVEIRA, T. C., 2014b, "Towards Facilities to Introduce Solutions for MDE in Development Environments with Reusable Assets". In: *International Conference on Information Reuse and Integration*, IRI'14, pp. 195–202, b.

BASSO, F. P., WERNER, C. M. L., OLIVEIRA, T. C., 2015a, "A Summary of Challenges for "MDE as Service"". In: *9th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems, At Belo Horizonte-MG, Brazil, Volume: 1*, WDES'15, pp. 85–88, a.

BASSO, F. P., OLIVEIRA, T. C., WERNER, C. M., et al., 2016a, "Analysis of Asset Specification Languages for Representation of Descriptive Data from {MDE} Artifacts", *Procedia Computer Science*, v. 100, pp. 221 – 228.

BASSO, F. P., WERNER, C. M. L., OLIVEIRA, T. C., et al., 2016b, "Criteria for Description of MDE Artifacts". In: *X Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems, At Maringá-PR, Brazil, Volume: 1*, WDES'16, pp. 80–84, b.

BASSO, F. P., 2006, *Features-Oriented Model-Driven Architecture: Uma Abordagem para MDD.* Master's thesis. Available at: <http://meriva.pucrs.br:8080/dspace/handle/10923/1520>.

BASSO, F. P., 2015, "A Proposal for a Common Representation Language for MDE Artifacts and Settings". In: *Doctoral Symposium at Software Technologies: Applications and Foundations 2015 Conference (STAF 2015), L'Aquila, Italy, July 20, 2015.*, pp. 21–31. Available at: <http://ceur-ws.org/Vol-1499/paper3.pdf>.

BASSO, F. P., 2016, "Student Research Abstract: MDE as Service, Overview and Research Progress". In: *31st ACM/SIGAPP Symposium on Applied Computing*, SAC'16, pp. 1586–1587.

BASSO, F. P., BECKER, L. B., OLIVEIRA, T. C., 2006, "Using the FOMDA approach to support object-oriented real-time systems development". In: *Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing*, ISORC 2006, pp. 374–381. doi: 10.1109/ISORC.2006.76.

BASSO, F. P., BECKER, L. B., OLIVEIRA, T. C., 2007, "Uma Solução para Reuso e Manutenção de Transformadores de Modelos Usando a Abordagem FOMDA". In: *Simpósio Brasileiro de Engenharia de Software. Anais do 21o Simpósio Brasileiro de Engenharia de Software.*, SBES 2007, pp. 130–146.

BASSO, F. P., BASSO, R. M. P., OLIVEIRA, T. C., et al., 2009, "FOMDA ML: A Language to Specify Model Transformers in MDA, Paper In Portuguese(FOMDA ML: Uma Linguagem para Especificação de Transformadores de Modelos na MDA)". In: *Third Rapid Application Development Workshop*, WDRA 2009.

BASSO, F. P., BASSO, R. M. P., OLIVEIRA, T. C., 2012, "Towards a Web Modeling Environment for a Model Driven Engineering Approach". In: *Third Brazilian Workshop on Model Driven Development*, BW-MDD 2012.

BASSO, F. P., OLIVEIRA, T. C., FARIAS, K., 2014c, "Extending JUnit 4 with Java Annotations and Reflection to Test Variant Model Transformation

Assets". In: *29th Symposium On Applied Computing*, SAC'14, pp. 1601–1608, c.

BASSO, F. P., PILLAT, R. M., FRANTZ, R. Z., et al., 2014d, "Assisted Tasks to Generate Pre-prototypes for Web Information Systems". In: *16th International Conference on Enterprise Information Systems.*, ICEIS'14, pp. 14–25, d.

BASSO, F. P., PILLAT, R. M., OLIVEIRA, T. C., et al., 2014e, "Generative Adaptation of Model Transformation Assets: Experiences, Lessons and Drawbacks". In: *29th Symposium On Applied Computing*, SAC'14, pp. 1027–1034, e.

BASSO, F. P., PILLAT, R. M., ROOZ-FRANTZ, F., et al., 2014f, "Study on Combining Model-Driven Engineering and Scrum to Produce Web Information Systems". In: *16th International Conference on Enterprise Information Systems*, ICEIS'14, pp. 137–144, f.

BASSO, F. P., PILLAT, R. M., ROOS-FRANTZ, F., et al., 2015b, "Combining MDE and Scrum on the Rapid Prototyping of Web Information Systems", *International Journal of Web Engineering and Technology*, v. 10, n. 3, pp. 214–244.

BASSO, F. P., OLIVEIRA, T. C., WERNER, C. M. L., et al., 2016c, "Analysis of Asset Specification Languages for Representation of Descriptive Data from MDE Artifacts". In: *International Conference on {ENTERprise} Information Systems/International Conference on Project MANagement/International Conference on Health and Social Care Information Systems and Technologies, CENTERIS/ProjMAN / {HCist} 2016, October 5-7*, CENTERIS'16, c.

BASSO, F. P., PILLAT, R. M., OLIVEIRA, T. C., et al., 2016d, "Automated design of multi-layered web information systems", *Journal of Systems and Software*, v. 117, pp. 612 – 637. ISSN: 0164-1212.

BASSO, F. P., OLIVEIRA, T. C., WERNER, C. M., et al., 2017a, "Building the foundations for 'MDE as Service'", *IET Software*, v. 11 (August), pp. 195–206(11).

BASSO, F. P., WERNER, C. M. L., DE OLIVEIRA, T. C., 2017b, "Automated Approach for Asset Integration in Eclipse IDE". In: *2017 IEEE/ACM Joint 5th International Workshop on Software Engineering for Systems-of-Systems and 11th Workshop on Distributed Software Development,*

Software Ecosystems and Systems-of-Systems, JSOSICSE, Buenos Aires, Argentina, May 23, 2017, pp. 34–40, b.

BASSO, F. P., WERNER, C. M. L., DE OLIVEIRA, T. C., 2017c, "Revisiting Criteria for Description of MDE Artifacts". In: *2017 IEEE/ACM Joint 5th International Workshop on Software Engineering for Systems-of-Systems and 11th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems, JSOSICSE, Buenos Aires, Argentina, May 23, 2017*, pp. 27–33, c.

BATORY, D., AZANZA, M., AO SARAIVA, J., 2008, "The Objects and Arrows of Computational Design". In: *Model Driven Engineering Languages and Systems*, pp. 1–20. doi: 10.1007/978-3-540-87875-9_1.

BATORY, D., GONCALVES, R., MARKER, B., et al., 2013a, "Dark Knowledge and Graph Grammars in Automated Software Design". In: *Invited Speaker at Software Language Engineering*, SLE'13, pp. 1–18, a. Available at: <http://www.cs.utexas.edu/ftp/predator/13SLE.pdf>.

BATORY, D., LATIMER, E., AZANZA, M., 2013b, "Teaching Model Driven Engineering from a Relational Database Perspective". In: *16th International Conference on Model Driven Engineering Languages and Systems*, MODELS'13, pp. 121–137, b. doi: 10.1007/978-3-642-41533-3_8.

BECKER, J., JANIESCH, C., PFEIFFER, D., 2007, "Situational Method Engineering: Fundamentals and Experiences: IFIP WG 8.1 Working Conference, 12–14 September 2007, Geneva, Switzerland". cap. Reuse Mechanisms in Situational Method Engineering, pp. 79–93, Boston, MA, Springer US. ISBN: 978-0-387-73947-2.

BECKER, L., HOLTZ, R., PEREIRA, C., 2002, "On mapping RT-UML specifications to RT-Java API: bridging the gap". In: *Object-Oriented Real-Time Distributed Computing, 2002. (ISORC 2002).*, pp. 348–355. doi: 10.1109/ISORC.2002.1003772.

BENAVIDES, D., SEGURA, S., RUIZ-CORTÉS, A., 2010, "Automated analysis of feature models 20 years later: A literature review", *Inf. Syst.*, v. 35, n. 6 (set.), pp. 615–636. ISSN: 0306-4379. doi: 10.1016/j.is.2010.01.001.

BENDRAOU, R., DESFRAY, P., GERVAIS, M.-P., et al., 2008, "MDA Tool Components: a proposal for packaging know-how in model driven

development", *Software & Systems Modeling*, v. 7, n. 3, pp. 329–343. ISSN: 1619-1366. doi: 10.1007/s10270-007-0058-8. Available at: <http://dx.doi.org/10.1007/s10270-007-0058-8>.

BERGER, T., RUBLACK, R., NAIR, D., et al., 2013, "A survey of variability modeling in industrial practice". In: *Seventh International Workshop on Variability Modelling of Software-intensive Systems*, VaMoS '13, pp. 1–7. doi: 10.1145/2430502.2430513.

BÉZIVIN, J., 2005, "On the unification power of models", *Software and System Modeling*, v. 4, n. 2, pp. 171–188. doi: 10.1007/s10270-005-0079-0.

BÉZIVIN, J., HAMMOUDI, S., LOPES, D., et al., 2004, "Applying MDA Approach for Web Service Platform". In: *Enterprise Distributed Object Computing Conference*, pp. 58–70. doi: 10.1109/EDOC.2004.10019.

BIEHL, M., EL-KHOURY, J., LOIRET, F., et al., 2011, "A Domain Specific Language for Generating Tool Integration Solutions". In: *4th Workshop on Model-Driven Tool & Process Integration (MDTPI)*.

BIEHL, M., EL-KHOURY, J., LOIRET, F., et al., 2014, "On the modeling and generation of service-oriented tool chains", *Software & Systems Modeling*, v. 13, n. 2, pp. 461–480. ISSN: 1619-1366. doi: 10.1007/s10270-012-0275-7. Available at: <http://dx.doi.org/10.1007/s10270-012-0275-7>.

BLANKENHORN, K., 2004, *A UML Profile for GUI Layout*. Master's thesis. Available at: <http://www.bitfolge.de/pubs/thesis/>.

BLOIS, A. P. T., LUMERTZ, P., OLIVEIRA, T. C., 2009, "Quantools: A MDA transformation approach", *Revista de Informtica Terica e Aplicada (Impresso)*, v. 16, pp. 53–56.

BOEHM, B., 2006, "A View of 20th and 21st Century Software Engineering". In: *28th International Conference on Software Engineering*, ICSE '06, pp. 12–29. doi: 10.1145/1134285.1134288.

BOOCH, G., RUMBAUGH, J., JACOBSON, I., 2005, *The Unified Modeling Language User Guide (2nd Edition)*. Addison-Wesley.

BOSCH, J., 2013, "Achieving Simplicity with the Three-Layer Product Model", *IEEE Computer*, v. 46, n. 11, pp. 34–39.

BOSCH, J., 2009, "From Software Product Lines to Software Ecosystems". In: *13th International Software Product Line Conference*, SPLC '09, pp. 111–119.

BOUNCKEN, R. B., GAST, J., KRAUS, S., et al., 2015, "Coopetition: a systematic review, synthesis, and future research directions", *Review of Managerial Science*, v. 9, n. 3 (Jul), pp. 577–601.

BRAMBILLA, M., FRATERNALI, P., 2014, "Large-scale Model-Driven Engineering of web user interaction: The WebML and WebRatio experience", *Science of Computer Programming*, v. 89, Part B, pp. 71 – 87. doi: http://dx.doi.org/10.1016/j.scico.2013.03.010. Special issue on Success Stories in Model Driven Engineering.

BRIAND, L., FALESSI, D., NEJATI, S., et al., 2012, "Research-based innovation: A tale of three projects in model-driven engineering". In: *15th International Conference on Model Driven Engineering Languages and Systems*, v. 7590, *MODELS 2012*, pp. 793–809.

BROWN, J. R., FEHIGE, Y., 2017, "Thought Experiments". In: Zalta, E. N. (Ed.), *The Stanford Encyclopedia of Philosophy*, summer 2017 ed., Metaphysics Research Lab, Stanford University.

BRUNELIÈRE, H., CABOT, J., CLASEN, C., et al., 2010, "Towards Model Driven Tool Interoperability: Bridging Eclipse and Microsoft Modeling Tools". In: *Modelling Foundations and Applications*, v. 6138, *Lecture Notes in Computer Science*, pp. 32–47. Springer Berlin Heidelberg. ISBN: 978-3-642-13594-1.

BURKE, B., MONSON-HAEFEL, R., 2006, *Enterprise JavaBeans 3.0: Developing Enterprise Java Components*. O'Reilly.

CAPILLA, R., BOSCH, J., TRINIDAD, P., et al., 2014, "An overview of Dynamic Software Product Line architectures and techniques: Observations from research and industry", *Journal of Systems and Software*, v. 91, n. 0, pp. 3–23. doi: http://dx.doi.org/10.1016/j.jss.2013.12.038.

CAPLAT, G., SOURROUILLE, J.-L., 2005, "Model Mapping Using Formalism Extensions", *IEEE Software*, v. 22, n. 2, pp. 44–51. ISSN: 0740-7459. doi: http://doi.ieeecomputersociety.org/10.1109/MS.2005.45.

CASTELLANOS, C., BORDE, E., PAUTET, L., et al., 2014, "Automatic Production of Transformation Chains Using Structural Constraints on Output Models". In: *Software Engineering and Advanced Applications (SEAA), 2014 40th EUROMICRO Conference on*, pp. 158–165, Aug. doi: 10.1109/SEAA.2014.13.

CASTELLUCCIA, D., BOFFOLI, N., 2014, "Service-oriented Product Lines: A Systematic Mapping Study", *SIGSOFT Softw. Eng. Notes*, v. 39, n. 2 (mar.), pp. 1–6.

CHOWDHURY, A. F., HUDA, M. N., 2011, "Comparison Between Adaptive Software Development and Feature Driven Development". In: *Computer Science and Network Technology (ICCSNT), 2011 International Conference on*, pp. 363–367. doi: 10.1109/ICCSNT.2011.6181977.

COMBEMALE, B., DEANTONI, J., BAUDRY, B., et al., 2014, "Globalizing Modeling Languages", *IEEE Computer, Institute of Electrical and Electronics Engineers*, v. 47, n. 6 (June), pp. 68–71.

COMBEMALE, B., CHENG, B. H., FRANCE, R. B., et al., 2015a, *Globalizing Domain-Specific Languages. International Dagstuhl Seminar, Dagstuhl Castle, Germany, October 5-10, 2014, Revised Papers*. Springer International Publishing.

COMBEMALE, B., CHENG, B. H., FRANCE, R. B., et al., 2015b, *Globalizing Domain-Specific Languages*. Springer International Publishing.

COOK, S., JONES, G., KENT, S., et al., 2007, *Domain-specific Development with Visual Studio Dsl Tools*. Addison-Wesley Professional. ISBN: 9780321398208.

CORRÊA, C., OLIVEIRA, T., WERNER, C., 2013, "Towards Coupled Evolution of Metamodels, Models, Graph-Based Transformations and Traceability Links". In: *25th International Conference on Software Engineering and Knowledge Engineering, SEKE 2013, Boston, USA, June 27-29 2013*, pp. 721–726.

CORREIA, T. P., GUESSI, M., DE OLIVEIRA, L. B. R., et al., 2016, "RARep: a Reference Architecture Repository". In: *The 28th International Conference on Software Engineering and Knowledge Engineering, SEKE 2016, Redwood City, San Francisco Bay, USA, July 1-3, 2016.*, pp. 363–368.

CRIADO, J., MARTÍNEZ, S., IRIBARNE, L., et al., 2015, "Enabling the reuse of stored model transformations through annotations". In: *International Conference on Model Transformations*, pp. 1–15, July.

CUADRADO, J. S., GUERRA, E., LARA, J. D., 2014, "A Component Model for Model Transformations", *IEEE Transactions on Software Engineering*, v. 40, n. 11 (Nov), pp. 1042–1060. doi: 10.1109/TSE.2014.2339852.

CUADRADO, J. S., GUERRA, E., DE LARA, J., 2015, "Reusable Model Transformation Components with bentō". In: *International Conference on Theory and Practice of Model Transformations, ICMT 2015*, pp. 59–65.

CUNHA, J., AO PAULO FERNANDES, J., MARTINS, P., et al., 2016, "Evaluating refactorings for spreadsheet models", *Journal of Systems and Software*, v. 118, pp. 234 – 250.

CZARNECKI, K., 2005, "Overview of generative software development". In: *2004 international conference on Unconventional Programming Paradigms*, UPP'04, pp. 326–341.

CZARNECKI, K., HELSEN, S., 2003, "Classification of Model Transformation Approaches". In: *OOPSLA'03 Workshop on Generative Techniques in the Context of Model-Driven Architecture*.

DA SILVA, A. R., 2015, "Model-driven engineering: A survey supported by the unified conceptual model", *Computer Languages, Systems & Structures*, v. 43, pp. 139 – 155. ISSN: 1477-8424.

D'AMBROS, M., GALL, H., LANZA, M., et al., 2008, "Analysing Software Repositories to Understand Software Evolution". In: *Software Evolution*, pp. 37–67, Berlin, Heidelberg, Springer Berlin Heidelberg.

DE LARA, J., VANGHELUWE, H., 2002, "Using AToM3 as a meta-case tool." In: *4th International Conference on Enterprise Information Systems (ICEIS)*, pp. 642–649.

DI RUSCIO, D., MALAVOLTA, I., MUCCINI, H., et al., 2012, "Model-Driven Techniques to Enhance Architectural Languages Interoperability". In: *Fundamental Approaches to Software Engineering*, FASE'12, pp. 26–42.

DOS SANTOS, R. P., WERNER, C., 2010, "Analyzing the Concept of Components in the Brechó-VCM Approach through a Sociotechnical and Software Reuse Management Perspective". In: *Software Components, Architectures and Reuse (SBCARS), 2010 Fourth Brazilian Symposium on*, pp. 21–30. doi: 10.1109/SBCARS.2010.26.

DOS SANTOS, R. P., ESTEVES, M. G. P., DE S. FREITAS, G., et al., 2013, "Software Ecosystems Comprehension and Evolution", *Social Networking*, v. 3, n. 2 (Feb), pp. 108–118.

DOS SANTOS ROCHA, R., FANTINATO, M., 2013, "The use of software product lines for business process management: A systematic literature review", *Information and Software Technology*, v. 55, n. 8, pp. 1355 – 1373.

ELAASAR, M., NEAL, A., 2013, "Integrating Modeling Tools in the Development Lifecycle with OSLC: A Case Study". In: *16th International Conference on Model Driven Engineering Languages and Systems*, MODELS'13, pp. 154–169.

ELGEDAWY, I., 2009, "Reusable SOA Assets Identification Using E-Business Patterns". In: *Services - II, 2009. SERVICES-2 '09. World Conference on*, pp. 33–40. doi: 10.1109/SERVICES-2.2009.10.

ELIAS, G., SCHUENCK, M., NEGÓCIO, Y., et al., 2006, "X-ARM: an asset representation model for component repository systems". In: *2006 ACM symposium on Applied computing*, SAC '06, pp. 1690–1694. ISBN: 1-59593-108-2. doi: 10.1145/1141277.1141676.

ELKOUTBI, M., KHRISS, I., KELLER, R. K., 2006, "Automated Prototyping of User Interfaces Based on UML Scenarios", *Automated Software Engg.*, v. 13, n. 1 (jan.), pp. 5–40. ISSN: 0928-8910. doi: 10.1007/s10515-006-5465-5. Available at: <http://dx.doi.org/10.1007/s10515-006-5465-5>.

ETIEN, A., ARANEGA, V., BLANC, X., et al., 2012, "Chaining model transformations". In: *First Workshop on the Analysis of Model Transformations*, AMT '12, pp. 9–14. ISBN: 978-1-4503-1803-7. doi: 10.1145/2432497.2432500. Available at: <http://doi.acm.org/10.1145/2432497.2432500>.

ETIEN, A., MULLER, A., LEGRAND, T., et al., 2013, "Localized model transformations for building large-scale transformations", *Software & Systems Modeling*, pp. 1–25. ISSN: 1619-1366. doi: 10.1007/s10270-013-0379-8.

FARIAS, K., 2010, "Empirical Evaluation of Effort on Composing Design Models". In: *32Nd ACM/IEEE International Conference on Software Engineering - Volume 2*, ICSE '10, pp. 405–408.

FARIAS, K., GARCIA, A., LUCENA, C., et al., 2014, "Towards a Quality Model for Model Composition Effort". In: *29th Symposium On Applied Computing*, SAC'14, pp. 1181–1183.

FERNANDES, P., WERNER, C., TEIXEIRA, E., 2011, "An Approach for Feature Modeling of Context-Aware Software Product Line", *Journal of Universal Computer Science*, v. 17, n. 5 (mar), pp. 807–829.

FORWARD, A., BADREDDIN, O., LETHBRIDGE, T., et al., 2012, "Model-driven rapid prototyping with Umple", *Software: Practice and Experience*, v. 42, n. 7, pp. 781–797. doi: 10.1002/spe.1155.

FRANCE, R., BIEMAN, J., CHENG, B., 2007, "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, pp. 311–317.

FRANCE, R., BIEMAN, J., CHENG, B. H., 2006, "CRI Collaborative Project Report: Repository for Model-Driven Development (ReMoDD)". In: *CRI of PI Meeting*, pp. 27–31.

FRANCE, R. B., BIEMAN, J. M., 2001, "Multi-View Software Evolution: A UML-based Framework for Evolving Object-Oriented Software". In: *ICSM*, pp. 386–395. doi: 10.1109/ICSM.2001.972751.

FRANTZ, R. Z., CORCHUELO, R., 2012, "Guaraná - Enterprise Application DSL and SDK Method & Tools", *Method & Tools*, v. 20, n. 1, pp. 59–64.

FUGGETTA, A., NITTO, E. D., 2014, "Software Process". In: *36th International Conference on Software Engineering*, ICSE '14, pp. 1–12. doi: 10.1145/2593882.2593883.

GAMMA, E., HELM, R., JOHNSON, R., et al., 1995, *Design Patterns, Elements of Reusable Object-Oriented Software*. Addison-Wesley.

GARCÉS, K., VARA, J. M., JOUAULT, F., et al., 2014, "Adapting transformations to metamodel changes via external transformation composition", *Software & Systems Modeling*, v. 13, n. 2, pp. 789–806. ISSN: 1619-1366. doi: 10.1007/s10270-012-0297-1. Available at: <http://dx.doi.org/10.1007/s10270-012-0297-1>.

GARTNER, 2013, *IT Asset Management: It's All About Process.* Technical report, Gartner, Inc. Available at: <http://www.gartner.com/imagesrv/media-products/pdf/provance/provance_issue1.pdf>.

GONÇALES, L. J., FARIAS, K., SCHOLL, M., et al., 2015, "Comparison of Design Models: A Systematic Mapping Study", *International Journal of Software Engineering and Knowledge Engineering*, v. 25, n. 09n10, pp. 1765–1769.

GORP, P. V., GREFEN, P. W. P. J., 2012, "Supporting the internet-based evaluation of research software with cloud infrastructure", *Software and System Modeling*, v. 11, n. 1, pp. 11–28.

GORP, P. V., MAZANEK, S., 2011, "SHARE: a web portal for creating and sharing executable research papers", *Procedia Computer Science*, v. 4, pp. 589 – 597. ISSN: 1877-0509. International Conference on Computational Science, ICCS 2011.

GORP, P. V., KELLER, A., JANSSENS, D., 2009, "Transformation Language Integration Based on Profiles and Higher Order Transformations". In: *Software Language Engineering*, v. 5452, *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 208–226. ISBN: 978-3-642-00433-9. doi: 10.1007/978-3-642-00434-6_14. Available at: <http://dx.doi.org/10.1007/978-3-642-00434-6_14>.

GREENFIELD, J., SHORT, K., 2003, "Software Factories: Assembling Applications with Patterns, Models, Frameworks and Tools". In: *Companion of the 18th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, OOPSLA '03, pp. 16–27. ISBN: 1-58113-751-6. doi: 10.1145/949344.949348.

GRUHN, V., 2002, "Process-Centered Software Engineering Environments, A Brief History and Future Challenges", *Ann. Softw. Eng.*, v. 14, n. 1-4 (dez.), pp. 363–382. ISSN: 1022-7091. doi: 10.1023/A:1020522111961.

GUOJIE, J., BAOLIN, Y., 2009, "An Encapsulation Structure and Description Specification for Application Level Software Components". In: *2009 International Conference on Computer Engineering and Technology*, v. 1, pp. 195–199, Jan.

GUY, C., COMBEMALE, B., DERRIEN, S., et al., 2012, "On Model Subtyping". In: *Modelling Foundations and Applications*, ECMFA 2012, pp. 400–415. doi: 10.1007/978-3-642-31491-9_30.

HADJI, H. B., SU-KYOUNG, K., HO-JIN, C., 2008, "A Representation Model for Reusable Assets to Support User Context". In: *IEEE International Symposium on Service-Oriented System Engineering, SOSE '08*, pp. 91–96.

HAILPERN, B., TARR, P. L., 2006, "Model-driven development: The good, the bad, and the ugly", *IBM Systems Journal*, v. 45, n. 3, pp. 451–462. doi: 10.1147/sj.453.0451.

HAMMOUDA, I., 2005, "A Tool Infrastructure for Model-Driven Development Using Aspectual Patterns". In: Beydeda, S., Book, M., Gruhn, V. (Eds.), *Model-Driven Software Development*, pp. 139–178. Springer Berlin Heidelberg. ISBN: 978-3-540-25613-7. doi: 10.1007/3-540-28554-7_7.

HEBIG, R., GABRYSIAK, G., GIESE, H., 2012, "Towards patterns for MDE-related processes to detect and handle changeability risks". In: *Software and System Process (ICSSP), 2012 International Conference on*, pp. 38–47, June. doi: 10.1109/ICSSP.2012.6225978.

HEBIG, R., GIESE, H., STALLMANN, F., et al., 2013, "On the Complex Nature of MDE Evolution". In: *Model Driven Engineering Languages and Systems*, MODELS 2013, pp. 436–453. doi: 10.1007/978-3-642-41533-3_27.

HEBIG, R., 2014, *Evolution of Model-Driven Engineering Settings in Practice*. PhD thesis, University of Potsdam, June. Available at: <http://opus.kobv.de/ubp/volltexte/2014/7076/>.

HEBIG, R., BENDRAOU, R., 2014, "On the Need to Study the Impact of Model Driven Engineering on Software Processes". In: *2014 International Conference on Software and System Process*, ICSSP 2014, pp. 164–168. doi: 10.1145/2600821.2600846.

HEIJSTEK, W., KUHNE, T., CHAUDRON, M., 2011, "Experimental Analysis of Textual and Graphical Representations for Software Architecture Design". In: *Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on*, pp. 167–176. doi: 10.1109/ESEM.2011.25.

HERMERSCHMIDT, L., HÖLLDOBLER, K., RUMPE, B., et al., 2013, "Generating Domain-Specific Transformation Languages for Component & Connector Architecture Descriptions". In: *Workshop on Model-Driven Engineering for Component-Based Software Systems*, (ModComp)'2015, pp. 18–23.

HOLDSCHICK, H., 2012, "Challenges in the Evolution of Model-based Software Product Lines in the Automotive Domain". In: *4th International Workshop on Feature-Oriented Software Development*, FOSD '12, pp. 70–73. doi: 10.1145/2377816.2377826.

HONG-MIN, R., ZHI-YING, Y., JING-ZHOU, Z., 2009, "Design and Implementation of RAS-Based Open Source Software Repository". In: *Sixth International Conference on Fuzzy Systems and Knowledge Discovery, 2009. FSKD '09.*, v. 2, pp. 219–223. doi: 10.1109/FSKD.2009.778.

HONG-MIN, R., JIN, L., JING-ZHOU, Z., 2010, "Software asset repository open framework supporting customizable faceted classification". In: *IEEE International Conference on Software Engineering and Service Sciences (ICSESS), 16-18 July, 2010*, pp. 1–4.

HORVÁTH, Á., VARRÓ, D., VARRÓ, G., 2006, "Automatic generation of platform-specific transformation", *Info-Communications-Technology*, v. LXI, n. 7, pp. 40–45.

HUTCHINSON, J., WHITTLE, J., ROUNCEFIELD, M., et al., 2011, "Empirical assessment of MDE in industry". In: *33rd International Conference on Software Engineering*, ICSE '11, pp. 471–480. ISBN: 978-1-4503-0445-0. doi: 10.1145/1985793.1985858.

ISO/IEC, 2011, *Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models*. Technical report, International Organization for Standardization - ISO/IEC 25010:2011. Available at: <`https://www.iso.org/standard/35733.html`>.

IZQUIERDO, J. L. C., CABOT, J., 2013, "Discovering Implicit Schemas in JSON Data". In: *13th International Conference on Web Engineering*, ICWE'13, pp. 68–83.

JACOBSON, I., NG, P.-W., MCMAHON, P. E., et al., 2012, "The Essence of Software Engineering: The SEMAT Kernel. A Thinking Framework in the Form of an Actionable Kernel." *ACMQUEUE. Development 9. Networks*, v. 10, n. 10, pp. 1–12.

JAKUMEIT, E., BUCHWALD, S., WAGELAAR, D., et al., 2014, "A survey and comparison of transformation tools based on the transformation tool contest", *Science of Computer Programming*, v. 85, Part A, n. 0, pp. 41 – 99. doi: http://dx.doi.org/10.1016/j.scico.2013.10.009.

JANSEN, S., FINKELSTEIN, A., BRINKKEMPER, S., 2009, "A sense of community: A research agenda for software ecosystems". In: *Software Engineering - Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on*, pp. 187–190. doi: 10.1109/ICSE-COMPANION.2009.5070978.

JERONIMO JR., H., WERNER, C., 2015, "A Systematic Mapping on the Relations between Systems-of-Systems and Software Ecosystems". In: *IX Workshop de Desenvolvimento Distribuído de Software, Ecossistemas de Software*

e Sistemas de Sistemas (WDES)/ VI Congresso Brasileiro de Software: Teoria e Prática, Brazil, Belo Horizonte, september, pp. 65–72.

JOHNSON, P., EKSTEDT, M., JACOBSON, I., 2012, "Where's the Theory for Software Engineering?" *Software, IEEE*, v. 29, n. 5, pp. 96–96. doi: 10.1109/MS.2012.127.

JOUAULT, F., VANHOOFF, B., BRUNELIÈRE, H., et al., 2010, "Inter-DSL coordination support by combining megamodeling and model weaving". In: *SAC'10*, pp. 2011–2018.

KANG, K., COHEN, S., HESS, J., et al., 1990, *Feature-Oriented Domain Analysis (FODA) Feasibility Study (CMU/SEI-90-TR-021)*. Software Engineering Institute, Carnegie Mellon University. Available at: <`http://www.sei.cmu.edu/library/abstracts/reports/90tr021.cfm`>.

KANG, K. C., KIM, S., LEE, J., et al., 1998, "FORM: A feature-oriented reuse method with domain-specific reference architectures", *Ann. Softw. Eng.*, v. 5 (jan.), pp. 143–168. ISSN: 1022-7091.

KAVALDJIAN, S., 2007, "A model-driven approach to generating user interfaces". In: *The 6th Joint Meeting on European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering: companion papers*, pp. 603–606. doi: 10.1145/1295014.1295053.

KELLY, S., TOLVANEN, J.-P., 2008, *Domain Specific Modeling: Enabling Full Code Generation*. IEEE Computer Society - John Wiley & Sons.

KENT, S., 2002, "Model Driven Engineering". In: *Integrated Formal Methods*, pp. 286–298. doi: 10.1007/3-540-47884-1_16.

KERSTEN, M., 2013, *Software Lifecycle Integration Architecture*. Technical report, Tasktop. Available at: <`http://www.tasktop.com`>.

KLEPPE, A., WARMER, J., BAST, W., 2003. "MDA Explained: The Model Driven Architecture: Practice and Promise". .

KOLOVOS, D. S., PAIGE, R. F., POLACK, F. A., 2006, "The Epsilon Object Language (EOL)". In: *Model Driven Architecture Foundations and Applications*, pp. 128–142. doi: 10.1007/11787044_11.

KOLOVOS, D., PAIGE, R., POLACK, F., 208, "The Epsilon Transformation Language". In: *International Conference on Model Transformation*, ICMT 2008, pp. 46–60.

KÖNEMANN, P., KINDLER, E., UNLAND, L., 2009, "Difference-based Model Synchronization in an Industrial MDD Process". In: *Second European Workshop on Model Driven Tool and Process Integration*, pp. 1–12, Fraunhofer Institute for Open Communication Systems.

KOUHEN, A. E., DUMOULIN, C., GERARD, S., et al., 2012, *Evaluation of Modeling Tools Adaptation*. Technical report, jan. Available at: <https://hal.archives-ouvertes.fr/hal-00706701>.

KRAUS, A., 2007, *Model Driven Software Engineering for Web Applications. Av at <http://nbn-resolving.de/urn:nbn:de:bvb:19-79362>*. PhD thesis, July. Available at: <http://nbn-resolving.de/urn:nbn:de:bvb:19-79362>.

KRIPPENDORFF, K., 2010, *"Combinatorial Explosion". Web Dictionary of Cybernetics and Systems*. PRINCIPIA CYBERNETICA WEB. Retrieved 29 November.

KROLL, P., MACISAAC, B., 2006, *Agility and Discipline Made Easy: Practices from OpenUP and RUP*. Addison-Wesley Professional.

KRUEGER, C. W., 1992, "Software reuse", *ACM Comput. Surv.*, v. 24, n. 2 (jun.), pp. 131–183. doi: 10.1145/130844.130856.

KULKARNI, V., BARAT, S., RAMTEERTHKAR, U., 2011, "Early experience with agile methodology in a model-driven approach". In: *Model-Driven Engineering Languages and Systems*, p. 578–590.

KURTEV, I., BÉZIVIN, J., JOUAULT, F., et al., 2006, "Model-based DSL frameworks". In: *OOPSLA Companion*, pp. 602–616. doi: 10.1145/1176617.1176632.

KUSEL, A., SCHÖNBÖCK, J., WIMMER, M., et al., 2015, "Reuse in model-to-model transformation languages: are we there yet?" *Software & Systems Modeling*, v. 14, n. 2, pp. 537–572. ISSN: 1619-1366. doi: 10.1007/s10270-013-0343-7.

KÜSTER, J. M., GSCHWIND, T., ZIMMERMANN, O., 2009, "Incremental Development of Model Transformation Chains Using Automated Testing". In: *Model Driven Engineering Languages and Systems*, v. 5795, *Lecture Notes in Computer Science*, pp. 733–747. ISBN: 978-3-642-04424-3.

LAFI, L., HAMMOUDI, S., FEKI, J., 2011, "Metamodel Matching Techniques in MDA: Challenge, Issues and Comparison". In: *Model and Data Engineering*, v. 6918, *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 278–286. doi: 10.1007/978-3-642-24443-8_29.

LAHTINEN, S., PELTONEN, J., HAMMOUDA, I., et al., 2006, "Guided Model Creation: A Task-Driven Approach". In: *Visual Languages and Human-Centric Computing, 2006. VL/HCC 2006. IEEE Symposium on*, pp. 89–94. doi: 10.1109/VLHCC.2006.25.

LANO, K., RAHIMI, S. K., 2014, "Model-Transformation Design Patterns", *IEEE Trans. Software Eng.*, v. 40, n. 12, pp. 1224–1259.

LANO, K., RAHIMI, S. K., TEHRANI, S. Y., et al., 2017, "A Survey of Model Transformation Design Pattern Usage". In: *Theory and Practice of Model Transformation - 10th International Conference, ICMT 2017, Held as Part of STAF 2017, Marburg, Germany, July 17-18, 2017*, pp. 108–118.

LEMOS, O. A. L., MASIERO, P. C., 2011, "A Pointcut-based Coverage Analysis Approach for Aspect-oriented Programs", *Inf. Sci.*, v. 181, n. 13 (jul.), pp. 2721–2746. ISSN: 0020-0255. doi: 10.1016/j.ins.2010.06.003.

LEVENDOVSZKY, T., LENGYEL, L., MEZEI, G., et al., 2005, "A Systematic Approach to Metamodeling Environments and Model Transformation Systems in VMTS", *Electron. Notes Theor. Comput. Sci.*, v. 127, n. 1 (mar.), pp. 65–75. ISSN: 1571-0661. doi: 10.1016/j.entcs.2004.12.040. Available at: <http://dx.doi.org/10.1016/j.entcs.2004.12.040>.

LIEBEL, G., MARKO, N., TICHY, M., et al., 2014, "Assessing the State-of-Practice of Model-Based Engineering in the Embedded Systems Domain". In: *Model-Driven Engineering Languages and Systems*, MODELS'14, pp. 166–182.

LIMA, T., DOS SANTOS, R. P., OLIVEIRA, J., et al., 2016, "The importance of socio-technical resources for software ecosystems management", *Journal of Innovation in Digital Ecosystems*, v. 3, n. 2, pp. 98 – 113.

LININGTON, P. F., 2005, "Automating support for e-business contracts", *Int. J. Cooperative Inf. Syst.*, v. 14, n. 2-3, pp. 77–98. doi: 10.1142/S0218843005001122.

LONIEWSKI, G., INSFRAN, E., ABRAHAO, S., 2010, "A systematic review of the use of requirements engineering techniques in model-driven development".

In: *13th international conference on Model driven engineering languages and systems: Part II*, MODELS'10, pp. 213–227.

LOPEZ-HERREJON, R. E., EGYED, A., TRUJILLO, S., et al., 2010, "Using Incremental Consistency Management for Conformance Checking in Feature-Oriented Model-Driven Engineering". In: *VaMoS'10*, pp. 93–100.

LUCAS, E. M., OLIVEIRA, T. C., FARIAS, K., et al., 2017, "CollabRDL: A language to coordinate collaborative reuse", *Journal of Systems and Software*, v. 131, pp. 505 – 527.

LÚCIO, L., MUSTAFIZ, S., DENIL, J., et al., 2013, "FTG+PM: An Integrated Framework for Investigating Model Transformation Chains". In: Khendek, F., Toeroe, M., Gherbi, A., et al. (Eds.), *SDL 2013: Model-Driven Dependability Engineering*, v. 7916, *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 182–202. ISBN: 978-3-642-38910-8. doi: 10.1007/978-3-642-38911-5_11. Available at: <http://dx.doi.org/10.1007/978-3-642-38911-5_11>.

LÚCIO, L., AMRANI, M., DINGEL, J., et al., 2014, "Model transformation intents and their properties", *Software & Systems Modeling*, pp. 1–38. ISSN: 1619-1366. doi: 10.1007/s10270-014-0429-x.

MACIEL, R. S. P., GOMES, R. A., MAGALHÃES, A. P. F., et al., 2013, "Supporting model-driven development using a process-centered software engineering environment", *Autom. Softw. Eng.*, v. 20, n. 3, pp. 427–461.

MADSEN, E. V., 2008, *Enterprise Requirements Management - using Rational Asset Manager*. PhD thesis, Technical University of Denmark, Informatics and Mathematical Modelling, May. Available at: <http://etd.dtu.dk/thesis/220259/dip08_23.pdf>.

MAGDALENO, A. M., DE OLIVEIRA BARROS, M., WERNER, C. M. L., et al., 2015, "Collaboration optimization in software process composition", *Journal of Systems and Software*, v. 103, pp. 452–466.

MANIKAS, K., 2016, "Revisiting software ecosystems Research: A longitudinal literature study", *Journal of Systems and Software*, v. 117, pp. 84 – 103.

MANIKAS, K., HANSEN, K. M., 2013, "Software ecosystems - A systematic literature review", *Journal of Systems and Software*, v. 86, n. 5, pp. 1294 – 1306.

MARIE FAVRE, J., NGUYEN, T., 2004, "Towards a Megamodel to Model Software Evolution through Transformations". In: *SETRA Workshop, Elsevier ENCTS*, pp. 59–74.

MARTINEZ-FERNANDEZ, S., SANTOS, P. S. M. D., AYALA, C. P., et al., 2015, "Aggregating Empirical Evidence about the Benefits and Drawbacks of Software Reference Architectures". In: *2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pp. 1–10, Oct.

MARTINEZ-RUIZ, T., GARCIA, F., PIATTINI, M., et al., 2011, "Applying AOSE Concepts to Model Crosscutting Variability in Variant-Rich Processes". In: *Software Engineering and Advanced Applications (SEAA), 2011 37th EUROMICRO Conference on*, pp. 334–338. doi: 10.1109/SEAA.2011.58.

MASCARENHAS, A. F. M., ANDRADE, A., MACIEL, R. P., 2013, "MTP: Model Transformation Profile". In: *Software Components, Architectures and Reuse (SBCARS), 2013 VII Brazilian Symposium on*, pp. 109–118, Sept. doi: 10.1109/SBCARS.2013.22.

MATINNEJAD, R., RAMSIN, R., 2012, "An Analytical Review of Process-Centered Software Engineering Environments". In: *Engineering of Computer Based Systems (ECBS)*, pp. 64–73. doi: 10.1109/ECBS.2012.11.

MELLOR, S. J., 2002, "Make Models Be Assets", *Commun. ACM*, v. 45, n. 11 (nov.), pp. 76–78. ISSN: 0001-0782. doi: 10.1145/581571.581597. Available at: <http://doi.acm.org/10.1145/581571.581597>.

MENGERINK, J., SCHIFFELERS, R. R. H., SEREBRENIK, A., et al., 2016, "DSL/Model Co-Evolution in Industrial EMF-Based MDSE Ecosystems". ME@MODELS, pp. 2–7.

MENS, T., GORP, P. V., 2006, "A Taxonomy of Model Transformation", *Electr. Notes Theor. Comput. Sci.*, v. 152, pp. 125–142. doi: 10.1016/j.entcs.2005.10.021.

MILLER, J., MUKERJI, J., 2003. "MDA Guide Version 1.0.1. Av. at <http://www.omg.org/cgi-bin/doc?omg/03-06-01>". Available at: <http://www.omg.org/cgi-bin/doc?omg/03-06-01>.

MOE, N. B., DINGSOYR, T., DYBA, T., 2010, "A teamwork model for understanding an agile team: A case study of a Scrum project", *Information and Software Technology*, v. 52, n. 5, pp. 480 – 491.

MOHAGHEGHI, P., AAGEDAL, J., 2007, "Evaluating Quality in Model-Driven Engineering". In: *International Workshop on Modeling in Software Engineering (MISE'07: ICSE Workshop 2007)*, pp. 6–6, May.

MOHAGHEGHI, P., 2008, "Evaluating Software Development Methodologies Based on their Practices and Promises". In: *New Trends in Software Methodologies, Tools and Techniques*, pp. 14–35.

MOHAGHEGHI, P., DEHLEN, V., 2008, "Where Is the Proof? - A Review of Experiences from Applying MDE in Industry". In: Schieferdecker, I., Hartman, A. (Eds.), *Model Driven Architecture – Foundations and Applications: 4th European Conference, ECMDA-FA 2008, Berlin, Germany, June 9-13, 2008.*, pp. 432–443, Berlin, Heidelberg, Springer Berlin Heidelberg.

MOHAGHEGHI, P., FERNANDEZ, M. A., MARTELL, J. A., et al., 2009, "MDE Adoption in Industry: Challenges and Success Criteria". In: Chaudron, M. R. (Ed.), *Models in Software Engineering*, v. 5421, *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 54–59. ISBN: 978-3-642-01647-9. doi: 10.1007/978-3-642-01648-6_6.

MOHAGHEGHI, P., GILANI, W., STEFANESCU, A., et al., 2013, "Where does model-driven engineering help? Experiences from three industrial cases." *Software & Systems Modeling*, v. 12, n. 3, pp. 619–639. ISSN: 1619-1374. doi: 10.1007/s10270-011-0219-7.

MOLINA, A. I., GIRALDO, W. J., GALLARDO, J., et al., 2012, "CIAT-GUI: A MDE-compliant environment for developing Graphical User Interfaces of information systems", *Advances in Engineering Software*, v. 52, pp. 10 – 29.

MONTEIRO, R., ASSUMPCAO PINEL, R., ZIMBRAO, G., et al., 2014a, "The MDArte experience: Organizational aspects acquired from a successful partnership between government and academia using model-driven development". In: *International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, pp. 575–586, Maya.

MONTEIRO, R., ZIMBRAO, G., MOREIRA DE SOUZA, J., 2014b, "Collaborative evolution process in MDArte: Exchanging solutions for information systems development among projects". In: *Computer Supported Cooperative Work in Design (CSCWD), 2014 IEEE 18th International Conference on*, pp. 569–574, Mayb. doi: 10.1109/CSCWD.2014.6846907.

MOTTA, R. C., DE OLIVEIRA, K. M., TRAVASSOS, G. H., 2017, "Rethinking Interoperability in Contemporary Software Systems". In: *Joint 5th International Workshop on Software Engineering for Systems-of-Systems and 11th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems*, JSOS '17, pp. 9–15.

MUSSBACHER, G., AMYOT, D., BREU, R., et al., 2014, "The Relevance of Model-Driven Engineering Thirty Years from Now". In: *Model-Driven Engineering Languages and Systems*, pp. 183–200. doi: 10.1007/978-3-319-11653-2_12.

NAKAGAWA, E. Y., OQUENDO, F., AVGERIOU, P., et al., 2015, "Foreword: Towards Reference Architectures for Systems-of-Systems". In: *3rd IEEE/ACM International Workshop on Software Engineering for Systems-of-Systems, SESoS 2015, Florence, Italy, May 17, 2015*, pp. 1–4.

NETO, A. C. D., SUBRAMANYAN, R., VIEIRA, M., et al., 2008, "Improving Evidence about Software Technologies: A Look at Model-Based Testing", *IEEE Software*, v. 25, n. 3, pp. 10–13.

NETO, V. V. G., GUESSI, M., OLIVEIRA, L. B. R., et al., 2014, "Investigating the Model-Driven Development for Systems-of-Systems". In: *ECSAW August 25 - 29, Vienna, Austria*, ECSAW'14, pp. –.

NEUBAUER, P., BERGMAYR, A., MAYERHOFER, T., et al., 2015, "XMLText: From XML Schema to Xtext". In: *2015 ACM SIGPLAN International Conference on Software Language Engineering*, SLE 2015, pp. 71–76.

NUNES, D. A., SCHWABE, D., 2006, "Rapid prototyping of web applications combining domain specific languages and model driven design". In: *6th international conference on Web engineering*, pp. 153–160. doi: 10.1145/1145581.1145616.

OGAWA, R. T., MALEN, B., 1991, "Towards Rigor in Reviews of Multivocal Literatures: Applying the Exploratory Case Study Method", *Review of Educational Research*, v. 61, n. 3, pp. 265–286.

OLIVEIRA, T. C., ALENCAR, P., COWAN, D., 2011, "ReuseTool-An extensible tool support for object-oriented framework reuse", *J. Syst. Softw.*, v. 84, n. 12 (dez.), pp. 2234–2252. ISSN: 0164-1212. doi: 10.1016/j.jss.2011.06.030.

OLIVEIRA JUNIOR, E. A., PAZIN, M. G., GIMENES, I. M. S., et al., 2013, "Product-Focused Software Process Improvement: 14th International Conference, PROFES 2013, Paphos, Cyprus, June 12-14, 2013." cap. SMartySPEM: A SPEM-Based Approach for Variability Management in Software Process Lines, pp. 169–183, Berlin, Heidelberg, Springer Berlin Heidelberg.

OMG, 2015, *Essence – Kernel and Language for Software Engineering Methods Version 1.1, SMSC/15-12-02.* Technical report, Object Management Group. Available at: <http://www.omg.org/spec/Essence/1.1/PDF>.

OMG, 2013, *UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems.* Technical report, Object Management Group. Available at: <www.omg.org/spec/MARTE/>.

OMG, 2008, *MOF Model to Text Transformation Language Version 1.0.* Technical report, Object Management Group. Available at: <http://www.omg.org/spec/MOFM2T08/1.0/>.

OMG, 2005, *RAS Reusable Asset Specification Version 2.2 November 2005. At September 2017.* Technical report, Object Management Group. Available at: <http://www.omg.org/spec/RAS/>.

OMG, 2002, *UML Profile For Schedulability, Performance, And Time.* Technical report, Object Management Group. Available at: <http://www.omg.org/spec/SPTP/>.

OSLC, 2017a, *OSLC Primer Web Page.* Technical report, Open Services for Lifecycle Collaboration, a. Available at: <open-services.net/resources/tutorials/oslc-primer/>.

OSLC, 2017b, *OSLC Articles - OSLC Lifecycle integration inspired by the web.* Technical report, Open Services for Lyfecicle Collaboration, b. Available at: <http://open-services.net/resources/articles/>.

PALMQUIST, M., 2014, *The Amazon Model: If You Can't Beat 'Em, Work with 'Em.* Technical report, Strategy+Business: Corporate Strategies and News Articles on Global Business, Management, Competition and Marketing. Available at: <https://www.strategy-business.com/blog/The-Amazon-Model-If-You-Cant-Beat-Em-Work-with-Em?gko=d33d3>.

PALUDO, M., REINEHR, S., MALUCELLI, A., et al., 2011, "Applying pattern structures to document and reuse components in component-based soft-

ware engineering environments". In: *2011 IEEE International Conference on Information Reuse Integration*, pp. 378–383, Aug.

PANDIT, N. R., 1996, *The creation of theory: A recent application of the grounded theory method*. The qualitative report, 2(4), 1-14.

PARK, S., PARK, S., SUGUMARAN, V., 2007, "Extending reusable asset specification to improve software reuse". In: *2007 ACM symposium on Applied computing*, SAC '07, pp. 1473–1478. ISBN: 1-59593-480-4. doi: 10.1145/1244002.1244317. Available at: <http://doi.acm.org/10.1145/1244002.1244317>.

PAULON, A., FROHLICH, A., BECKER, L., et al., 2013, "Model-Driven Development of WSN Applications". In: *III Brazilian Symposium on Computing Systems Engineering (SBESC), At Niterói, RJ, Brazil*, pp. 161–166. doi: 10.1109/SBESC.2013.27.

PAULON, A., FROHLICH, A., BECKER, L., et al., 2014, "Wireless Sensor Network UML Profile to Support Model-Driven Development". In: *12th IEEE International Conference on Industrial Informatics, At Porto Alegre, RS, Brazil*, INDIN 2014, pp. 227–232. doi: 10.1109/INDIN.2014.6945512.

PEFFERS, K., TUUNANEN, T., ROTHENBERGER, M., et al., 2007, "A Design Science Research Methodology for Information Systems Research", *J. Manage. Inf. Syst.*, v. 24, n. 3 (dez.), pp. 45–77. ISSN: 0742-1222. doi: 10.2753/MIS0742-1222240302.

PETERSEN, K., FELDT, R., MUJTABA, S., et al., 2008, "Systematic Mapping Studies in Software Engineering". In: *12th International Conference on Evaluation and Assessment in Software Engineering*, EASE'08, pp. 68–77.

PETRE, M., 2013, "UML in practice". In: *2013 International Conference on Software Engineering*, ICSE '13, pp. 722–731. ISBN: 978-1-4673-3076-3.

PILLAT, R. M., BASSO, F. P., OLIVEIRA, T. C., et al., 2013, "Ensuring consistency of feature-based decisions with a business rule system". In: *Seventh International Workshop on Variability Modelling of Software-intensive Systems*, VaMoS '13, pp. 15:1–15:8. ISBN: 978-1-4503-1541-8.

PILLAT, R. M., OLIVEIRA, T. C., ALENCAR, P. S., et al., 2015, "BPMNt: A BPMN extension for specifying software process tailoring", *Information and Software Technology*, v. 57, n. 0, pp. 95 – 115. doi: http://dx.doi.org/10.1016/j.infsof.2014.09.004.

POLGÁR, B., RÁTH, I., SZATMARI, Z., et al., 2009, "Model-based Integration, Execution and Certification of Development Tool-chains". In: *2th Workshop on Model-Driven Tool & Process Integration (MDTPI)*, pp. 35–46.

QUERCINI, G., REYNAUD, C., 2013, "Entity discovery and annotation in tables". In: *16th International Conference on Extending Database Technology*, EDBT '13, pp. 693–704. ISBN: 978-1-4503-1597-5. doi: 10.1145/2452376.2452457.

RESCHENHOFER, T., MATTHES, F., 2015, "An Empirical Study on Spreadsheet Shortcomings from an Information Systems Perspective". In: *Business Information Systems - 18th International Conference, BIS 2015, Poznań, Poland, June 24-26, 2015*, pp. 50–61.

RISTIĆ, S., LUKOVIĆ, I., ALEKSIĆ, S., et al., 2012, "An approach to the specification of user interface templates for business applications". In: *Fifth Balkan Conference in Informatics*, pp. 124–129. doi: 10.1145/2371316.2371340.

RITALA, P., GOLNAM, A., WEGMANN, A., 2014, "Coopetition-based business models: The case of Amazon.com", *Industrial Marketing Management*, v. 43, n. 2, pp. 236 – 249.

RIVERA, J. E., RUIZ-GONZALEZ, D., LOPEZ-ROMERO, F., et al., 2009, "Orchestrating ATL model transformations". In: *Eur. Conf. Model. Found. Appl.*, pp. 34–46.

ROCCO, J. D., RUSCIO, D. D., IOVINO, L., et al., 2015, "Collaborative Repositories in Model-Driven Engineering [Software Technology]", *Software, IEEE*, v. 32, n. 3 (May), pp. 28–34. ISSN: 0740-7459. doi: 10.1109/MS.2015.61.

ROCCO, J. D., RUSCIO, D. D., IOVINO, L., et al., 2014, "Dealing with the Coupled Evolution of Metamodels and Model-to-text Transformations". In: *International Workshop on Workshop on Models and Evolution*, ME@MODELS, pp. 22–31.

ROCCO, J. D., RUSCIO, D. D., PIERANTONIO, A., et al., 2016, "Using ATL Transformation Services in the MDEForge Collaborative Modeling Platform". In: *Theory and Practice of Model Transformations - 9th International Conference, ICMT 2016, Held as Part of STAF 2016, Vienna, Austria, July 4-5, 2016*, pp. 70–78.

ROSE, L., GUERRA, E., LARA, J., et al., 2013, "Genericity for model management operations", *Software & Systems Modeling*, v. 12, n. 1, pp. 201–219. doi: 10.1007/s10270-011-0203-2.

ROTHENBERG, J., 1989, "Artificial Intelligence, Simulation &Amp; Modeling". John Wiley & Sons, Inc., cap. The Nature of Modeling, pp. 75–92, New York, NY, USA. ISBN: 0-471-60599-9. Available at: <http://dl.acm.org/citation.cfm?id=73119.73122>.

ROYCHOUDHURY, S., GRAY, J., JOUAULT, F., 2011, "A Model-Driven Framework for Aspect Weaver Construction". In: *Transactions on Aspect-Oriented Software Development VIII*, v. 6580, *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 1–45. ISBN: 978-3-642-22030-2. doi: 10.1007/978-3-642-22031-9_1. Available at: <http://dx.doi.org/10.1007/978-3-642-22031-9_1>.

RUNESON, P., HÖST, M., 2008, "Guidelines for conducting and reporting case study research in software engineering", *Empirical Software Engineering*, v. 14, n. 2, pp. 131.

SANTOS, R., WERNER, C., TOSTES, L., et al., 2016, "Supporting negotiation and socialization for component markets in software ecosystems context". In: *2016 XLII Latin American Computing Conference (CLEI)*, pp. 1–12.

SANTOS, R. P. D., WERNER, C., 2012, "ReuseECOS: An Approach to Support Global Software Development Through Software Ecosystems". ICGSEW, pp. 60–65, Washington, USA. IEEE.

SCHMIDT, D. C., 2006, "Guest Editor's Introduction: Model-Driven Engineering", *IEEE Computer*, v. 39, n. 2, pp. 25–31. doi: 10.1109/MC.2006.58.

SELIC, B., RUMBAUGH, J., 1998, *Using UML for Modelling Complex Real-Time Systems*. Technical report, Rational. Available at: <www.dcc.ttu.ee/LAP/ISP0011/umlrt.pdf>.

SELIC, B., 2005, "On Software Platforms, Their Modelling with UML 2, and Platform-Independent Design". In: *8th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, ISORC'05, pp. 15–21.

SELIC, B., 2006, "Model-driven development: its essence and opportunities". In: *Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing, 2006. ISORC 2006.*, pp. 7 pp.–. doi: 10.1109/ISORC.2006.54.

SELIC, B., 2014, "Model-Based Software Engineering in Industry: Revolution, Evolution, or Smoke?" In: *Invited Speaker at International Conference on Industrial Informatics. Av. at <http://www.fosslc.org/drupal/content/model-based-software-engineering-industry-revolution-evolution-or-smoke>. At Dec.*, pp. –.

SHAN, T. C., HUA, W. W., 2006, "Taxonomy of Java Web Application Frameworks". In: *2006 IEEE International Conference on e-Business Engineering (ICEBE 2006), 24-26 October 2006, Shanghai, China*, pp. 378–385.

SMOLANDER, K., ROSSI, M., PEKKOLA, S., 2017, "Infrastructures, Integration and Architecting During and After Digital Transformation". In: *Joint 5th International Workshop on Software Engineering for Systems-of-Systems and 11th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems*, JSOS '17, pp. 23–26.

SOMMERVILLE, I., 2010, *Software Engineering (9th Edition)*. Addison-Wesley.

SOUZA, V. E. S., FALBO, R. D. A., GUIZZARDI, G., 2007, "A UML Profile for Modeling Framework-based Web Information Systems". In: *12th International Workshop on Exploring Modelling Methods in Systems Analysis and Design EMMSAD2007*, pp. 153–162.

STARY, C., 2000, "Contextual prototyping of user interfaces". In: *3rd conference on Designing interactive systems: processes, practices, methods, and techniques*, pp. 388–395. doi: 10.1145/347642.347797.

STEINBERG, D., BUDINSKY, F., PATERNOSTRO, M., et al., 2008, *EMF: Eclipse Modeling Framework (2nd Edition)*. Addison-Wesley Professional.

STOCQ, J., VANDERDONCKT, J., 2004, "A domain model-driven approach for producing user interfaces to multi-platform information systems". In: *working conference on Advanced visual interfaces*, pp. 395–398. doi: 10.1145/989863.989934.

STRITTMATTER, M., HEINRICH, R., 2016, "Challenges in evolving Metamodels", *Softwaretechnik-Trends*, v. 36, n. 1.

STRUBER, D., SCHULZ, S., 2016, "A tool environment for managing families of model transformation rules". In: *9th International Conference on Graph Transformation, ICGT 2016 in Memory of Hartmut Ehrig held as part of Conference on Software Technologies: Applications and Foundations, STAF 2016; Vienna; Austria; 5 July 2016 through 6 July 2016*, v. 9761, pp. 89–101.

SYRIANI, E., VANGHELUWE, H., LASHOMB, B., 2015, "T-Core: a framework for custom-built model transformation engines", *Software & Systems Modeling*, v. 14, n. 3, pp. 1215–1243.

TEKINERDOĞAN, B., BILIR, S., ABATLEVI, C., 2005, "Integrating platform selection rules in the model driven architecture approach". In: *2003 European conference on Model Driven Architecture: foundations and Applications*, MDAFA'03, pp. 159–173. doi: 10.1007/11538097_11.

THÜM, T., APEL, S., KÄSTNER, C., et al., 2014, "A Classification and Survey of Analysis Strategies for Software Product Lines", *ACM Comput. Surv.*, v. 47, n. 1 (jun.), pp. 6:1–6:45. ISSN: 0360-0300. doi: 10.1145/2580950.

TOLVANEN, J.-P., 2016, "MetaEdit+ for Collaborative Language Engineering and Language Use (Tool Demo)". In: *2016 ACM SIGPLAN International Conference on Software Language Engineering*, SLE 2016, pp. 41–45.

TORCHIANO, M., TOMASSETTI, F., RICCA, F., et al., 2013, "Relevance, Benefits, and Problems of Software Modelling and Model Driven techniques-A Survey in the Italian Industry", *J. Syst. Softw.*, v. 86, n. 8 (aug), pp. 2110–2126. ISSN: 0164-1212. doi: 10.1016/j.jss.2013.03.084.

TRAN, H. N., COULETTE, B., TRAN, D. T., et al., 2011, "Automatic Reuse of Process Patterns in Process Modeling". In: *26th Symposium On Applied Computing*, SAC'11, pp. 1431–1438.

TURNER, D. A., CHAE, J., 2010, *Java Web Programming with Eclipse*. CreateSpace Independent Publishing Platform.

VALE, T., CRNKOVIC, I., DE ALMEIDA, E. S., et al., 2016, "Twenty-eight years of component-based software engineering", *Journal of Systems and Software*, v. 111, pp. 128–148.

VANDERDONCKT, J., 2005, "A MDA-compliant environment for developing user interfaces of information systems". In: *17th international conference on Advanced Information Systems Engineering*, pp. 16–31. doi: 10.1007/11431855_2.

VANHOOFF, B., BAELEN, S. V., HOVSEPYAN, A., et al., 2006, "Towards a transformation chain modeling language". In: *6th international conference on Embedded Computer Systems: architectures, Modeling, and Simulation*, SAMOS'06, pp. 39–48. doi: 10.1007/11796435_6.

VARA, J., BOLLATI, V., JIMÉNEZ, A., et al., 2014, "Dealing with Traceability in the MDD of Model Transformations", *Transactions on Software Engineering*, v. 40, n. 6, pp. 555–583. doi: 10.1109/TSE.2014.2316132.

VIGNAGA, A., JOUAULT, F., BASTARRICA, M. C., et al., 2013, "Typing artifacts in megamodeling", *Software & Systems Modeling*, v. 12, n. 1, pp. 105–119. ISSN: 1619-1366. doi: 10.1007/s10270-011-0191-2.

VOELTER, M., 2009, "Best Practices for DSLs and Model-Driven Development", *Journal of Object Technology*, v. 8, n. 6, pp. 79–102.

VÖELTER, M., GROHER, I., 2007, "Handling Variability in Model Transformations and Generators". In: *Companion to the Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2007)*, pp. 1–8. ACM. Available at: <http://www.voelter.de/data/workshops/HandlingVariability InModelTransformations.pdf>.

VÖLTER, M., VISSER, E., 2011, "Product Line Engineering Using Domain-Specific Languages". In: *Software Product Line Conference (SPLC), 2011 15th International*, pp. 70–79. doi: 10.1109/SPLC.2011.25.

WAGELAAR, D., 2006, "Blackbox Composition of Model Transformations using Domain-Specific Modelling Languages". In: *ECMDA Composition of Model Transformations Workshop*, pp. 15–19.

WASSERMAN, A. I., 1990, "Tool integration in software engineering environments". In: Long, F. (Ed.), *Software Engineering Environments*, v. 467, *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 137–149. ISBN: 978-3-540-53452-5. Available at: <http://dx.doi.org/10.1007/3-540-53452-0_38>.

WEIQING, Z., LEILDE, V., MOLLER-PEDERSEN, B., et al., 2012, "Towards Tool Integration through Artifacts and Roles". In: *Software Engineering Conference (APSEC), 2012 19th Asia-Pacific*, pp. 603–613. doi: 10.1109/APSEC.2012.45.

WERNER, C., MURTA, L., MARINHO, A., et al., 2009, "Towards a Component and Service Marketplace with Brechó Library". In: *IADIS International Conference on WWW/Internet, Rome, Italy, nov. 2009*, IADIS'09, pp. 567–574.

WHITTLE, J., HUTCHINSON, J., ROUNCEFIELD, M., et al., 2013, "Industrial Adoption of Model-Driven Engineering: Are the Tools Really the Problem?" In: *16th International Conference on Model Driven Engineering Languages and Systems*, MODELS'13, pp. 1–17.

WHITTLE, J., HUTCHINSON, J., ROUNCEFIELD, M., et al., 2015, "A taxonomy of tool-related issues affecting the adoption of model-driven engineering", *Software & Systems Modeling*, pp. 1–19. ISSN: 1619-1374.

WILLINK, E. D., 2003, "UMLX: A graphical transformation language for MDA". In: *Model-Driven Architecture: Foundations and Applications*, pp. 13–24.

WIMMER, M., MARTÍNEZ, S., JOUAULT, F., et al., 2012, "A Catalogue of Refactorings for Model-to-Model Transformations", *Journal of Object Technology*, v. 11, n. 2 (ago.), pp. 2:1–40. ISSN: 1660-1769. doi: 10.5381/jot.2012.11.2.a2.

YEATES, L., 2004, *Thought Experimentation: A Cognitive Approach*. Master's thesis. Available at: <`https://www.dropbox.com/s/nczwi1vkb7x88qd/TECA.pdf?dl=0`>.

YICK, J., MUKHERJEE, B., GHOSAL, D., 2008, "Wireless Sensor Network Survey", *Comput. Netw.*, v. 52, n. 12 (ago.), pp. 2292–2330. doi: 10.1016/j.comnet.2008.04.002.

YIE, A., CASALLAS, R., DERIDDER, D., et al., 2012, "Realizing Model Transformation Chain interoperability", *Software & Systems Modeling*, v. 11, n. 1, pp. 55–75. ISSN: 1619-1366. doi: 10.1007/s10270-010-0179-3.

YIN, R., 2003, *Case Study Research: Design and Methods*. SAGE Publications.

ZAKHEIM, B., 2017, "How Difficult Can It Be to Integrate Software Development Tools? The Hard Truth", *InfoQ*, (January). Available at: <`https://www.infoq.com/articles/tool-integration-hard-truth`>.

ZHANG, W., 2015, "Tool Integration by Models, Not Only by Metamodels - Applying Modeling to Tool Integration". In: *Model-Driven Engineering and Software Development, MODELSWARD*, pp. 461–469.

ZHANG, W., MOLLER-PEDERSEN, B., 2013, "Establishing tool chains above the service cloud with integration models". In: *IEEE 20th International Conference on Web Services, ICWS 2013*, pp. 372–379.

ZHANG, W., MOLLER-PEDERSEN, B., 2014, "Modeling of tool integration resources with OSLC support". In: *Model-Driven Engineering and Software Development (MODELSWARD), 2014 2nd International Conference on*, pp. 99–110, Jan.

ZHANG, W., MOLLER-PEDERSEN, B., BIEHL, M., 2012, "A Light-weight Tool Integration Approach - From a Tool Integration Model to OSLC Integration Services." In: *International Conference on Software Engineering and Applications, ICSOFT'12*, pp. 137–146.

ZHANG, Y., PATEL, S., 2011, "Agile Model-Driven Development in Practice", *Software, IEEE*, v. 28, n. 2 (March), pp. 84–91.

Appendix A

Complementary Material

> What is right is not always popular and what is popular is not always right.
>
> Albert Einstein

A.1 Highlights of the Academic Trajectory

A.1.1 Communications

We presented our contributions, all derived from the main research theme MDE as a Service, in means of communications shown in Table A.1. A total of 25 publications shown in Table A.2 reports my research effort since the starting of this PhD, complemented in Figure A.1 with statistical data. Figure A.1.A shows the number of publications, which are divided into:

1. Oral presentations, as shown in Figure A.1.B. This includes 20 papers (see also Figure A.1.C), distributed in a Doctoral Symposium, Student Research Competition (SRC), Workshops (only three are national workshops), Conferences (only one published nationally) and Book Chapters.

2. Periodicals (journals) shown in Figure A.1.D.

Figure A.1.E presents a time-line contextualizing these publications between 2012 and 2017.

A.1.2 Research Cooperation

In order to reach this point of contributions, we collaborated with some researchers in the area. These collaborations also tend to generate new contributions in RAS++

Table A.1: Means of communication

Title	Full Title	Type	Qualis	JCR
BWMDD	Brazilian Workshop on Model Driven Development	Workshop	-	-
CENTERIS	International Conference on Enterprise Information Systems	Conference	B3	-
GPCE	International Conference on Generative Programming: Concepts & Experiences	Conference	B2	-
ICEIS	International Conference on Enterprise Information Systems	Conference	B2	-
ICSR	International Conference on Software Reuse	Conference & Book Chapter	B2	-
IET-SW	IET Software	Journal	B1	0.733
INDIN	International Conference on Industrial Informatics	Conference	B1	-
IJWET	International Journal of Web Engineering and Technology	Journal	B1	RG 0.47
IRI	International Conference on Information Reuse and Integration	Conference	B1	-
JSS	Journal of Systems and Software	Journal	A2	1.245
PCS	Procedia Computer Science	Journal	C	RG 1.08
SBESC	Brazilian Symposium on Computing Systems Engineering	Conference	B2	-
SAC	Symposium on Applied Computing	Conference	A1	-
SEKE	International Conference on Software Engineering and Knowledge Engineering	Conference	B1	-
SESOS	International Workshop on Software Engineering for Systems-of-Systems	Workshop	B4	-
SIGPLAN	ACM SIGPLAN Notices	Journal	B3	0.621
SRC	Symposium on Applied Computing - SRC	Student Res. Competition	A1	-
STAF-DS	Software Technologies: Applications and Foundations	Doctoral Symposium	-	-
VaMoS	International Workshop on Variability Modelling of Software-intensive Systems	Conference	A2	-
WDES	Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems	Workshop	B5	-

Table A.2: List of publications

Id	Where?	Paper Title	First Author	Pages	Reference
P01	BWMDD	Towards a Web Modeling Environment for a Model Driven Engineering Approach	Yes	8	BASSO *et al.* (2012)
P02	VaMoS	Ensuring consistency of feature-based decisions with a business rule system	No	8	PILLAT *et al.* (2013)
P03	ICSR	A Common Representation for Reuse Assistants	Yes	6	BASSO *et al.* (2013b)
P04	SEKE	How do You Execute Reuse Tasks Among Tools?	Yes	6	BASSO *et al.* (2013c)
P05	GPCE	Supporting Large Scale Model Transformation Reuse	Yes	10	BASSO *et al.* (2013a)
P06	SBESC	Model-Driven Development of WSN Applications	No	6	PAULON *et al.* (2013)
P07	SGIPLAN	Newsletter - Supporting Large Scale Model Transformation Reuse	Yes	10	BASSO *et al.* (2014a)
P08	ICEIS	Study on Combining Model-Driven Engineering and Scrum to Produce Web Information Systems	Yes	8	BASSO *et al.* (2014f)
P09	ICEIS	Assisted Tasks to Generate Pre-prototypes for Web Information Systems	Yes	8	BASSO *et al.* (2014d)
P10	SAC	Generative Adaptation of Model Transformation Assets: Experiences, Lessons and Drawbacks	Yes	8	BASSO *et al.* (2014e)
P11	SAC	Extending JUnit 4 with Java Annotations and Reflection to Test Variant Model Transformation Assets	Yes	8	BASSO *et al.* (2014c)
P12	SAC	Towards a Quality Model for Model Composition Effort	No	3	FARIAS *et al.* (2014)
P13	IRI	Towards Facilities to Introduce Solutions for MDE in Development Environments with Reusable Assets	Yes	8	BASSO *et al.* (2014b)
P14	INDIN	Wireless Sensor Network UML Profile to Support Model-Driven Development	No	6	PAULON *et al.* (2014)
P15	STAF-DS	A Proposal for a Common Representation Language for MDE Artifacts and Settings	Yes	10	BASSO (2015)
P16	WDES	A Summary of Challenges for "MDE as Service"	Yes	4	BASSO *et al.* (2015a)
P17	IJWET	Combining MDE and Scrum on the Rapid Prototyping of Web Information Systems	Yes	30	BASSO *et al.* (2015a)
P18	JSS	Automated design of multi-layered web information systems	Yes	25	BASSO *et al.* (2016d)
P19	CENTERIS	Analysis of Asset Specification Languages for Representation of Descriptive Data from MDE Artifacts	Yes	8	BASSO *et al.* (2016c)
P20	PCS	Analysis of Asset Specification Languages for Representation of Descriptive Data from MDE Artifacts	Yes	8	BASSO *et al.* (2016a)
P21	WDES	Criteria for Description of MDE Artifacts	Yes	4	BASSO *et al.* (2016b)
P22	SRC	Student Research Abstract: MDE as Service, Overview and Research Progress	Yes	2	BASSO (2016)
P23	SESOS	Revisiting Criteria for Description of MDE Artifacts	Yes	7	BASSO *et al.* (2017c)
P24	SESOS	Automated Approach for Asset Integration in Eclipse IDE	Yes	7	BASSO *et al.* (2017b)
P25	IET-SW	Building the foundations for "MDE as Service"	Yes	11	BASSO *et al.* (2017a)

and MDE as a Service. Thus, the results from this thesis are also start points for future researches. In the following we highlight the main initiatives:

I have continuously interacted with researchers from UFRJ to discuss and validate ideas as follows: 1) prof. Dra. Cláudia M. L. Werner; 2) prof. Dr. Toacy C.

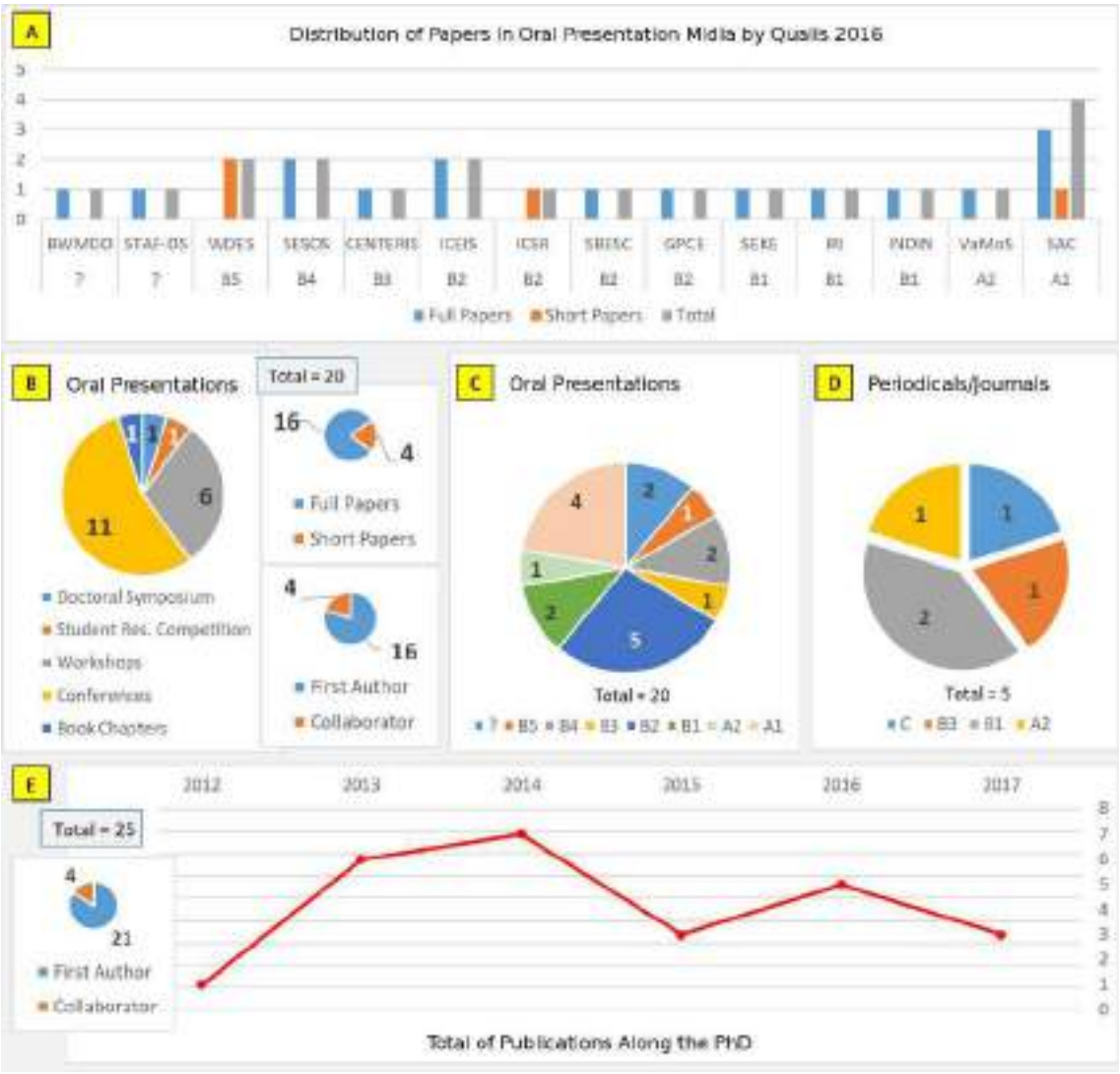


Figure A.1: Analysis of scientific production

Oliveira; 3) the PHD student Raquel M. Pillat, co-author of experience reports, and 4) with the research groups PRISMA and Reuse Group.

I helped on the research conducted by a Master student (André Ruza Paulon, UFSC, Advisor Leandro B. Becker) between 2012 and 1014. His dissertation was presented in December, 2014. It is titled "Desenvolvimento Dirigido a Modelos Para Redes de Sensores Sem Fio". My contribution is: 1) configuration of the tool support; 2) mentoring on the use and application of the FOMDA Approach; 3) evaluation of the conducted activities; and 4) Reporting the results as second author of his papers.

Recent collaborations derived from a workshop, WDES 2015, and include the following researchers: 1) Prof. Dr. Rodrigo Pereira dos Santos, from the Universidade Federal do Estado do Rio de Janeiro (Unirio) and; Valdemar Graciano Neto (UFG), under the supervision of Prof. Dra. Elisa Yumi Nakagawa at Universidade de São Paulo (USP). We already have one published result. We also conducted

meetings and wrote two other materials, to be finished in 2017, scoping the theme MDE Ecosystems.

Two mapping studies are result from collaborations with the Prof. Dr. Kleinner Farias from Unisinos. Three characterization studies of MDE as a Service are result from cooperation with prof. Dr Rafael Z. Frantz and Fabricia Rooz-frantz from Unijui.

I have also collaborated with international researchers. We are currently collaborating with prof. Vasco Amaral, assistant professor at the Department of Computer Science group Science and Technology of Programming Section (CTP) of UNL (Universidade Nova de Lisboa). We already have aligned two papers discussing MDE as a Service with data mined from software projects. Another collaboration is with the PhD student at King's College London under the supervision of Dr. Kevin Lano. This contact is result from an experience report in combining MDE and Agility, we made some meetings to discuss issues in MDE as a Service that will be considered on her thesis. Thus, our contributions have been considered relevant by international researchers too.

We have two trials of research cooperation that are suspended. I have interacted for few months with Chessman Corrêa, a PHD student from UFRJ. Chessman has the ability for discussing MDE in a technical level. I provided some data requested by him to conduct his research, packing it in a CD. I had also interacted with him in some meetings. Unfortunately, for personal reasons, he did not complete his PHD. Besides, aiming at evaluating RAS++ in a real implementation for MDE as a Service (the MDArte research group), a trial for research cooperation was started in 2015 through prof. Dr. Geraldo Zimbrão da Silva. Due to some reasons, the team was dissolved and it was not possible to run a planned experiment.

Finally, I have also presented works in conferences where I am not a co-author: 1) Presenting a work from Ivens Portugal (UFRJ) in SEKE 2013; 2) Presenting a work from Chessman Corrêa (UFRJ) in SEKE 2013; and 3) Presenting a work from prof. DR. Luis Alvaro Silva (UFSM) in IRI 2014.

A.1.3 Reviews

I have also reviewed some papers, strengthen our research groups in themes as software reuse and MDE. I co-reviewed three conference papers, one submitted for the BWMDD 2012, other for the ICSR conference in 2013 and other to SBES in 2017. I co-reviewed three papers from international journals, one related with MDE-SDP submitted for the J-UCS in 2012, other related with MTCs submitted for the "Revista Ingenieria e Investigacion" in 2013, and other recently submitted for the SQJ. I have become also a registered reviewer of JSS, with two complete reviews

(2016, 2017).

A.2 Follow-up

As consequence of the lack of explicit technical information about tools, ZAKHEIM (2017) claims that Software Engineers are still incurring in large costs for integration. The possibility of execution of automatic transformations before integration could benefit software engineers by reducing integration costs. This section summarizes some ongoing and future works that proposes some sort of facility preliminaries to the state of the art in integration/tool chains.

A.2.1 Ongoing Works

Considering complementary representations for RAS++, Table A.3 presents 6 technical reports including increments for RAS++. Each report is for different scenarios in MDE as a Service, introducing toolboxes that have been developed with different intents in support to preliminary phases for tool chain. Thus, in the following we present a summary for each report:

T01 - Reusable artifacts/resources for MDE are not wide spread in software development industry. Aiming at improving its diffusion, the research of the area recommends the use of at least two reuse approaches: 1) one is motivated in a global scale for reuse i.e., for inter-organizational contexts, whose mechanism is always structured on a collaborative repository through assets; 2) a second is adopted more locally by technicians, usually adopted for managing intra-organizational reuse, integrating these artifacts in flexible settings for customization. A challenge for modern reuse tools is to support the diffusion of artifacts using a common format. Thus, the report T01 introduces RAS++ tool prototypes as a mean to assist this diffusion.

T02 - This thesis focused in RAS++ as a way to represent assets in a common format, without attention to features from tool support developed along four years. In T02, we investigate whether features from tool support is relevant to surpass an issue pointed out by the literature: artifact repositories have been proposed in the last 10 years, however the lack of critical mass imposes obstacles for collaboration (ROCCO *et al.*, 2015). Accordingly, as suggest data mined from the ReMoDD repository, it is observed little interest from academy and industry in sharing their artifacts. Some recent researches claim that the lack of adequate platforms to support reuse is one of the reasons (COMBEMALE *et al.*, 2014; CRIADO *et al.*, 2015; DOS SANTOS *et al.*, 2013; LÚCIO *et al.*, 2014; ROCCO *et al.*, 2016). In this sense, since the level of technical details in assets requires a considerable effort for

representation, the lack of appropriate support for representation may be one of the reasons for the low interest too. As presented in our analysis of reuse costs, it is observed a big reuse cost for producing reusable assets conforming to RAS++. Thus, this report presents a possible solution for cost reduction through tool support for automatic representation of technicalities for assets.

Table A.3: Technical reports for RAS++ tool support

Id	Title	Available at
T01	Supporting the MDE Diffusion with RAS++ Prototypes	prisma.cos.ufrj.br/wct/tr01.pdf
T02	Prototypes for Representation of Artifacts and Settings in RAS++	prisma.cos.ufrj.br/wct/tr02.pdf
T03	Reuse of MDE Artifacts and Settings through RAS++	prisma.cos.ufrj.br/wct/tr03.pdf
T04	Federation of COTS for MDE With a Pivotal Representation Language	prisma.cos.ufrj.br/wct/tr04.pdf
T05	Pivoting Process Modeling Languages with RAS++	prisma.cos.ufrj.br/wct/tr05.pdf
T06	RAS++ Method Scoping Preliminary Phases for Tool Chain	prisma.cos.ufrj.br/wct/tr06.pdf
T07	Towards Integrating MDE Ecosystems with RAS++	prisma.cos.ufrj.br/wct/tr07.pdf

Id	Assessment Plan	Research Method	Artifact Dev. Contexts	Assessment Perspectives	Representation Languages
T01	Case Study/MSR	Exploratory	Academy	P03, P04, P09	RAS++
T02	Experiment	Exploratory	Academy	P03, P04, P09	RAS++
T03	Case Study/MSR	Exploratory	Academy	P03, P04, P09	RAS++ and FOMDA DSL
T04	Case Study/MSR	Exploratory	Academy	P03, P04, P09	RAS++
T05	Analytical Study	Exploratory	Academy	P03, P04, P09	RAS++ and PMLs
T06	Experiment	Exploratory	Academy	P03, P04, P09	RAS++
T07	Benchmark	Exploratory	Academy	P03, P04, P09	RAS++

T03 - Reuse of MDE components is handled by at least two reuse scopes: one is a mechanism introduced in a reuse repository and the other in reuse mechanisms established through MDE Artifacts and Settings. These scopes need a bridge, but the state of the art is unable to make this connection through a general representation with less coupled components. Through a case study, we have demonstrated that the scopes are well connectable through the use of model transformations, assist the conversion from assets represented with RAS++ to FOMDA DSL, a target representation in MDE Artifacts and Settings adopted in previous implementation of MDE as a Service. Thus, this study demonstrates the execution of the phase "Transformation", allowing the transition of an abstract representation, whit a generic purpose in MDE as a Service, to a concrete one adopted for tool chain purposes.

T04 - The development of Components-of-the-Shelf (COTS) for MDE Artifacts has been suggested as a way to reduce costs for the implementation of MDE-based processes. A difficulty for reusing COTS is that the state of the practice in MDE adopts several repositories to store these components, imposing difficulties to the advent of coopetition in the area. Federation is a concept used on cloud computing that demands the management of information associated with artifacts when stored in different repositories. This report represents a case study considering federation of assets represented with RAS++. So far, we observed that RAS++ is representative for federating MDE artifacts and settings in existing repositories, thus important for future implementations of coopetition in MDE as a Service.

T05 - Industry found difficulties to adopt MDE in terms of tool chain: it is hard to integrate third-party MDE Artifacts into their contexts. This report considers as target context some process model representations. In this study we are auto-

matically introducing, through model transformations (CUADRADO *et al.*, 2014) and tailoring rules (PILLAT *et al.*, 2015), MDE Artifacts in process models. We, therefore, are using four supporting tools: one to design reusable assets, other for generating model-to-model transformations, other for executions of transformations and another to integrate the generated model-pieces in BPMN models. A case study has proven to be promising, suggesting that RAS++ promotes some benefits for process engineering not allowed in state of the art.

T06 - In order to introduce MDE Tools in tool chains, we proposed the use of at least three phases: 1) Specification - search existing tools on the web, 2) Acquisition - download, analyze and compare tool features, 3) Transformation - adapt and integrate tools in target software production environments. It is important the conceptualization of an integrated engineering solution to integrate of these phases. T06 aims at depicting a methodology and associated engineering solution to introduce tools for MDE in target development environments through reusable assets represented with RAS++.

T07 - MDE as a Service is an emergent scenario on software development characterized by specialized services for the introduction of MDE in target contexts. From an ecosystems perspective, resources are analyzed, acquired from repositories, adapted and integrated in software development environments. This technical report introduces a pivotal representation language called RAS++, which add the ecosystems perspective in MDE as a Service. It is designed considering data represented in MDE Ecosystems such as ReMoDD, GEMOC and SEMAT. RAS++ is an extension for the Reusable Asset Specification (RAS), an OMG standard for components reuse, and Asset Management Specification (AMS), thus also in conformity with the standard Open Services for Lifecycle Collaboration (OSLC). Thus, we contribute for investigation of a new reuse perspective, more general and inter-disciplinary in software engineering and MDE.

A.2.2 Considerations for Future Implementations

In order to understand issues and benefits from existing approaches for coopetition, it is important to classify different technical issues presented by different implementation scenarios. For example, we can mention some technical differences for implementation of coopetition when implemented through Amazon web service platform (RITALA *et al.*, 2014) and in MDE as Service: 1) Amazon is a reference, so there is not need for other platform (a web service interface is the solution for book sale), while in MDE as Service the platforms are diverse (a web service would not fix the observed combinatorial issue); 2) properties from Amazon' assets (products, books, etc.) are well known, while properties from MDE Artifacts and Settings

are still not fully mapped (we extracted some from literature mappings focusing in tool chain), and; 3) we still do now know the best way to cooperate and sale Software Engineering services associated with MDE Artifacts, while Amazon is expert in business models.

In the following, we present our findings in considerations for implementations of coopetition in the future.

Evolution of RAS++

Considerations in the Specification phase: The literature of the area (COMBEMALE *et al.*, 2014) says that we have more questions than answers about the problematic discussed in this thesis. We agree. In our position, the implementation of coopetition is more complex than the one reported by (ROCCO *et al.*, 2015) concerning collaboration and even more complex than the use of RAS++. ROCCO *et al.* (2015) presented many interesting issues for collaboration in MDE repositories, but they are restricted to facilities introduced in the level of model management, query and persistence and technical implementation for acquisition of artifacts through interfaces for Software-as-Service.

Considerations in the Acquisition phase: RAS++ is in part also motivated in a context for collaboration, but considering also competition and a conceptual issue rather than an implementation for a repository. In this sense, we included studies in RAS++ for mapping the phase "Acquisition", which relates to platforms for negotiation and decision making. Negotiation and decision making are essential elements for coopetition and are not properly discussed by the investigated literature for MDE. Thus, open questions for future investigation include: 1) whether existing platforms for ecosystems fulfill the needs of collaborative and competitive services that require to introduce MDE in target contexts, and 2) what else is important for implementation of coopetition in MDE as Service than the criteria we have proposed in phase "Acquisition"?

Considerations in the Transformation phase: Another research gap regards the inclusion or not of concepts for software product lines, or SPL, in a pivotal language. RAS++ was built on metaclasses that are classified as common for tool chain, which means that some concepts from specific representations for MDE Settings are not included. For example, Bentõ DSL includes a Feature Model (CUADRADO *et al.*, 2014). Due to the existence of few proposals suggesting that the integration of SPL concepts in tool chain are relevant for the context of MDE Settings (only two, including ours (BASSO *et al.*, 2014e)), we considered SPL as "not essential" for assets. Thus, while a consensus is not reached, the inclusion of software product line concepts such as the Feature Model in RAS++ is an open question.

Considerations in our research groups: Properties introduced in RAS++

for the Specification and Acquisition phase tend to not present much changes along years, but this is not true for the Transformation phase. This is because we limited this thesis in tool chain properties. However, our research group has been working with many other representation elements that also need integration in some target context of MDE as Service, but that are ignored by this thesis due to its focus. For example, it is also interesting to find properties from Object-Oriented Framework Instantiation (OOFI) (OLIVEIRA *et al.*, 2011), Model-Driven Software Product Line (MDSPL) (FERNANDES *et al.*, 2011), Aspects (FARIAS, 2010) and other related with process engineering (MAGDALENO *et al.*, 2015; PILLAT *et al.*, 2015). We believe that, through an analysis of focus group concerning the Transformation phase, properties from DSLs proposed by these works can be introduced in RAS++ as well. This would need assessment focused in scenarios for implementation MDE as Service built on process engineering needs.

Considerations for the research community: Through an extensive mapping, we conclude that properties from Specification and Acquisition phases can be used as is, thus providing foundational properties for integration of other concepts than tool chain. These properties are built on standards: RAS, an important standard proposed by the OMG to structure elements for reuse through instructions of integration and classification of artifacts in repositories, and AMS, a modern industry standard for cloud repositories. This is a good reason for adoption of RAS++. In order to evaluate its limitations, it is needed a research effort for application of RAS++ in specific repositories and integration scenarios, thus opening a promising field of research for the entire research community.

Standardization

A common representation language is a benefit in itself to the research area: 1) in terms of conceptualization, it organizes several concepts that the literature is considering important for MDE Artifacts and Settings; 2) in terms of a knowledge base, it allows new researches to start from a stable notation for representation; and, 3) in terms of implementation of a global reuse scenario, a pivotal language can represent essential elements for coopetition in MDE as Service.

Finally, this is the right moment for the research to explore a common representation language. Reuse opportunities are evident, coopetition is a desirable feature and the research could investigate requirements for its implementation starting from the contributions in this thesis.

A.3 A Personal Perspective

This thesis introduced coopetition in MDE as a Service as a long-term goal for implementation. This required appropriated characterization, which has been carried out through many studies reported in this thesis. In order to reach a foundational language, we analyzed the literature of the area to find out common properties. This section complements these contributions with highlights for a position about business opportunity. This position is our last considerations for MDE as a Service, built on previous experiences in a real industrial initiative.

We have seen initiatives for innovation, successfully deployed in scientific and technical incubators from Brazil, through companies founded to provide specialized services for software tests and software quality assurance. These are software engineering topics whose demand for services increased since 2000. Many works have been reporting that MDE is good for quality, maintenance, and productivity on software development. Thus, the question is why not specialized services for MDE in a business perspective?

I had invested four years of my life, between 2007 and 2011, in an implementation effort for this emerging scenario in a start-up company called Adapit. It was developed a business plan for MDE as a Service, focusing in start-up contexts as candidate for introduction of MDE Artifacts and Settings. Along these years, it was possible to observe that start-up contexts are more interested in producing code as fast as possible, thus letting attributes from software such as quality and modularization associated with MDE in a second plan. Besides, we noticed that MDE as a Service has a low demand in comparison to others for testing and quality assurance. The considered market (software start-ups) was not the ideal for implementation of MDE as a Service. We have not seen a better moment now either. Thus, currently it is hard to focus on a business plan for MDE as a Service with a low demand.

One of the reasons that hampered the implementation of this opportunity by Adapit is the focus to just one market segment for MDE adoption, restricted for systems engineering for startup software factories. Meanwhile, the market associated with process engineering seems to be more promising due to the need for certification in maturity models. For this scenario in MDE as a Service, software engineers are forced to look for alternatives for improving quality, creating demand for this type of service. Thus, we conclude that the industry and academy still need to create needs for scenarios of systems engineering based on MDE, so that new initiatives prove to potential customers that it is worth the investment in this regard.

Our long-term expectation is to attract more interest for development practices that include MDE. Meanwhile, toolboxes and methodology are continually improved. In this sense, it is important to anticipate a reminder to those interested in the im-

plementation of MDE as a Service. Although considering prepared to perform these adaptation services, Adapit company failed to attract an expressive number of software houses in the Brazilian market. Others have reported much more success, but they provide services for contexts that present little or none need for adaptation (BRAMBILLA and FRATERNALI, 2014; MONTEIRO *et al.*, 2014a). This raises a question whether prospects for MDE adoption are willing to pay the price for adapting resources?

Few prospects are a barrier to implement MDE as a Service. However, it can be surpassed by fixing issues for adoption (TORCHIANO *et al.*, 2013; WHITTLE *et al.*, 2015), such as some for preliminary phases for tool chain that characterize an important contribution of this thesis. As the search progresses and the interest for adoption of MDE increases, we believe that new demands for customization of MDE Artifacts, coopetition and integration, will promote new initiatives for MDE as a Service soon. Besides, DSLs and technologies evolve over time, which will inevitably lead to the development of new or adoption of similar toolboxes presented in previous chapters. Thus, we conclude that contributions from this thesis will motivate improvements in business, tools and methods for MDE in the near future.