Implementations Supporting Automated Technology Transfer in MDE as a Service

Fábio Paulo Basso^{1,2}, Elder de Macedo Rodrigues^{1}, Maicon Bernardino^{1}, Cláudia Maria Lima Werner^{2}, Toacy Cavalcante de Oliveira^{2}

^{1}Laboratory of Empirical Studies in Software Engineering (LESSE),
Federal University of Pampa (UNIPAMPA), Alegrete, RS, Brazil

^{2}Systems Engineering and Computer Science Department,
Federal University of Rio de Janeiro, RJ, Brazil

{fabiobasso,elderrodrigues}@unipampa.edu.br,

bernardino@acm.org, {werner,toacy}@cos.ufrj.br

Abstract. *Coopetition characterizes scenarios where competing firms establish collaborations for some intent. Here applies foundational studies built on software reuse and asset standards, such as for OSLC and RAS, asset platforms, smart contracts and others that are essential for promoting coopetition in the area. Our contribution is an experience report that provide foundations for technology transfer to software factory coopetition scenarios. Our goal is to characterize some coopetition approaches and tool support assisting Software Engineering (SE) tasks for technology transfer. This paper presents an experience report analyzing multi-case studies, implemented in local scenarios for coopetition, characterizing the research area with a summary of the main findings. The presented results are twofold: 1) With the current tool support in the state-of-the art, automation of SE tasks for technology transfer is feasible to local scenarios; and 2) However, global coopetition scenarios impose new implementation barriers.*

1. Introduction

Coopetition is characterized as a mean for competitors increase their business opportunities [Ritala et al. 2014]. Coopetition is a reality to some companies such as Amazon, but yet few understood when applied to the Software Engineering body of knowledge. To the Software Engineering (SE) contexts, coopetition could promote services performed by many professionals with know how in specific software development phases, an interesting research topic debated recently in a round table promoted by the International Conference on Software Engineering (ICSE) 2017. Coopetition is the mean by which Software Engineering services could be promoted in blocks [Boehm 2006]. For instance, some phases can be automated through resources for Model Driven Engineering (MDE) [Fuggetta and Nitto 2014], which requires the execution of other Software Engineering activities to introduce resources such as tools in contexts for adoption [Liebel et al. 2014]. This execution can be assisted through the implementation of concepts for "Software Factory", a term introduced in 2003 by Greenfield [Greenfield and Short 2003] to classify approaches that compose and adapt these resources.

In previous experiences, blocks supporting the interests from a set of Software Factories are represented as a domain model. This representation combines structural features for Model Transformation Chain (MTC), Software Product Lines (SPL) and Component-Based Development (CBD) [Basso et al. 2017a]. Through the implementation of these concepts, flexible resources are created, composed and introduced in one or more contexts for MDE adoption within one or more coopetition blocks. In this sense, we defined “MDE as a Service” as a Software Engineering that requires the application of these concepts to promote coopetition through reuse of MDE assets in inter-organizational contexts [Monteiro et al. 2014, Basso et al. 2013a, Basso et al. 2015, Basso et al. 2016b]. As illustrated in the center of Figure 1, resources (core assets) are used by more than one target company for MDE adoption (Company A and B). We have implemented and adopted concepts for Software Factory in real scenarios [Basso et al. 2013a, Basso et al. 2017a, Basso et al. 2006].

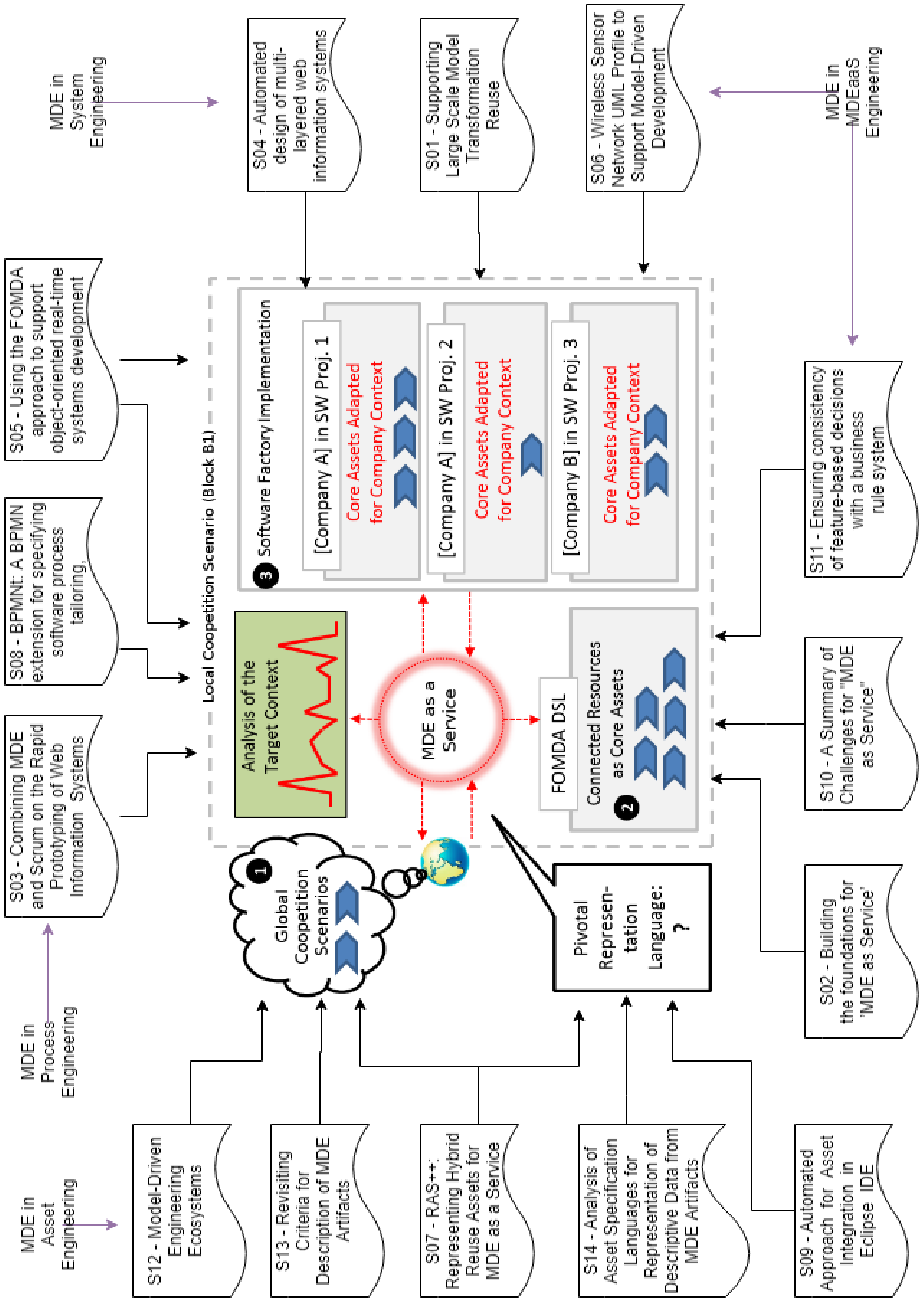
In this approach it is also important to consider the know how of teams to support the design and development tasks, which may imply on the execution of a feasibility study before introducing MDE in the target context. In [Basso et al. 2015] we discuss how we applied a feasibility study, adapting model-based tasks according to needs from an agile software project, which adopted Scrum as reference model for the Software Development Process (SDP).

This paper reports the implementation of the coopetition in Software Engineering field, providing a good evidence to the area. However, the new scenario shown in Figure 1 (box 1) is vaguely characterized, thus needing in-depth investigations for emergent scenarios. This work contributes for this characterization and it is organized as follows: As a characterization of experiences, Section 2 introduces emergent coopetition scenarios, complemented by Section 3 with open research topics and by Section 4 with concluding remarks.

2. Experiences on Implementations for Emergent Coopetition Scenarios

Through the FOMDA Approach [Basso et al. 2017a], resources developed for MDE (*e.g.*, model transformations, DSLs and tools) are used by different contexts, *i.e.* applied in many domains (or cross-application domains), which characterizes block of local coopetition.

This is possible due to the adaptation in existing resources for MDE such as MTCs and model transformations, delivering at the end a configured tool in conformity with target context. Our tool support includes the execution of generative and run-time/adaptive approaches. Three local coopetition blocks are successfully implemented [Basso et al. 2013a, Basso et al. 2017a, Basso et al. 2006], as illustrated by the groups of studies (S01, S05 and S06) which results in the following three domain models supporting MDE toolchain for Software Engineering companies: 1) Block B1 supporting the development of Web Information Systems; 2) Block B2 supporting the development of Embedded Real-time Systems; 3) Block B3 supporting the development of Wireless Sensor Networks for Internet of Things (IoT). What our previous experiences are suggesting is that local coopetition in Software Engineering is a reality, which means that we have the minimum tools and practices that support the implementation of the scenario shown in Figure 1 (box 2 and 3). Although some other examples of local coopetition are found in the literature too,



MDE in System Engineering
S05 - Using the FOMDA approach to support object-oriented real-time systems development
S08 - BPMNt: A BPMN extension for specifying software process tailoring.
S03 - Combining MDE and Scrum on the Rapid Prototyping of Web Information Systems
MDE in Process Engineering
MDE in Asset Engineering
S04 - Automated design of multi-layered web information systems
S01 - Supporting Large Scale Model Transformation Reuse
S06 - Wireless Sensor Network UML Profile to Support Model-Driven Development
MDE in MDEaaS Engineering
Local Coopetition Scenario (Block B1)
3 Software Factory Implementation
[Company A] in SW Proj. 1
Core Assets Adapted for Company Context
[Company A] in SW Proj. 2
Core Assets Adapted for Company Context
[Company B] in SW Proj. 3
Core Assets Adapted for Company Context
Analysis of the Target Context
MDE as a Service
FOMDA DSL
Connected Resources as Core Assets
2
1 Global Coopetition Scenarios
Pivotal Represen-tation Language: ?
S11 - Ensuring consistency of feature-based decisions with a business rule system
S10 - A Summary of Challenges for "MDE as Service"
S02 - Building the foundations for 'MDE as Service'
S12 - Model-Driven Engineering Ecosystems
S13 - Revisiting Criteria for Description of MDE Artifacts
S07 - RAS++: Representing Hybrid Reuse Assets for MDE as a Service
S14 - Analysis of Asset Specification Languages for Representation of Descriptive Data from MDE Artifacts
S09 - Automated Approach for Asset Integration in Eclipse IDE

Figura 1. Contribution Facet of Each Reported Experience in Scenarios for MDE as a Service.

and considering the surveys in the current body of knowledge [Zakheim 2017, Bouncken et al. 2015], global coopetition in Software Engineering is not a reality yet. In this sense, starting from the previous experiences in local coopetition to the MDE business, a recent PhD thesis (S07) investigated whether MDE as a Service could be implemented considering emergent global scenarios.

This research includes many characterization studies including experience reports, literature mappings and case studies, through which only recently we achieved a better understanding. Some of these contributions mapped the research area considering the state of the practice in MDE as a Service. We found reports suggesting that the MDE as a Service have been performed manually. Issues reported are associated with needs in tool support such as flexibility, integration of components, adaptation, test, and others. Thus, while many tools are available in support for Software Engineering activities associated with this approach, the state-of-practice execute MDE as a Service manually.

In other studies we focused on the reuse scope shown in Figure 1 (2 and 3), discussing different ways to implement systematic reuse through adaptations to specific contexts within system engineering (S01, S04, S05 and S06), and process engineering (S03, S05, and S08) engineering [Basso et al. 2017a, Pillat et al. 2015, Basso et al. 2017b]. For example, works in [Yie et al. 2012, Cuadrado et al. 2014] suggest the use of DSLs for MTC, SPL and CBD in order to assist Software Engineering activities, including fragmentation, composition, orchestration and test of MDE resources. While the state-of-the-practice reports the lack of appropriate tool support as an issue for the MDE adoption, the state-of-the-art presents some alternatives to use in this first scope of MDE as a Service. Therefore, we have found that the research on the implementation of software factory concepts for systematic reuse, which is a systematic reuse approach for MDE, is mature.

Recent studies detailed in S07 [Basso 2017] focused on the scope illustrated in Figure 1 (1): emergent scenarios with opportunities for global coopetition implemented through opportunistic reuse and design. It includes representations for MDE resources found in repositories and asset platforms. However, only few contributions are considering inter-organizational reuse, which imposes limits in approaches for managing coopetition scenarios [Bouncken et al. 2015]. So, we have found that the body of knowledge lacks requirements for implementation of global coopetition in SE. Moreover, although we have found a good availability of repositories promoting coopetition opportunities, except a research scoping collaborative design tools [Franzago et al. 2017], characterizing collaborative solutions for a totally different intent than the motivated in this paper, we have not found contributions focused in coopetition through asset engineering, which is an opportunistic reuse approach.

For this reason, the connection between these two reuse scopes for coopetition, local promoting systematic reuse and global promoting opportunistic reuse, is not yet well understood, remaining obfuscated. We found that all the related contributions in the Software Engineering base of knowledge present representations that can be used in one or other reuse scope for MDE as a Service. This means that one contribution can work well for part, but not all, the illustrated scenario in Figure 1. Due to the lack of a common/pivotal representation language, such scopes are not properly connected in existing contributions. Therefore, we investigated a novel DSL to solve this limitation, a differential that connects scopes from Figure 1 (1 and 2) [Basso 2017] through services built on assets.

In this sense, in [Basso et al. 2016a] we analyzed asset specification languages, concluding that the Reusable Asset Specification (RAS) is an interesting complement that can help on the implementation of collaboration through reuse repositories, but it is limited to provide descriptive information associated with MDE resources. So, we developed a preliminary version of RAS++ DSL and a tool prototype in [Basso et al. 2013b], adding to RAS metaclasses that we found as a common representation among many proposals for MTCs [Basso et al. 2014]. Moreover, in [Basso et al. 2013c] we presented a workflow that includes RAS++ DSL in this context. The implementation of this workflow will allow to automatically retrieve assets from reuse repositories and integrate them into the core assets managed by the "Software Factory".

These gaps derived form previous experiences are necessary to reach criteria to promote coopetition in Software Engineering and they are missed in the literature of the area. This limitation in the current wisdom become evident in the recent round-table promoted by ICSE 2017, where experts agreed that very few is acknowledged to draw any conclusion. As a long-term goal for the MDE community, some authors suggested that solutions for MDE should be shared on the web in a Knowledge Base (KB) [Mussbacher et al. 2014], thus allowing the quick discovery and comprehension of appropriate MDE resources (model transformations, tools, metamodels, files, etc.).

We then proposed RAS++ [Basso 2017], a DSL for asset engineering that introduces to the MDEaaS scenario the following features: 1) Support for extensibility, through meta-classes based on UML Profiles (*e.g.*, tags and stereotypes); 2) Support for representation of MDE Settings for toolchain as blocks of cooperation and coopetition, through meta-classes for a common representation language between MDE Settings and reuse repositories (*e.g.*, model typing); 3) Support for distributed/federated MDE Resources, through meta-classes for distributed information stored on the web (*e.g.*, deployment descriptors); 4) Support for distributed/federated Assets, through meta-classes for dependency among assets located in different repositories; 5) Support for service instantiation, through meta-classes for integration of Asset's content into target environments/DSLs for MDE Settings.

RAS++ is a DSL built on two standards: the Reusable Asset Specification (RAS) [Hong-min et al. 2010] and the Asset Management Specification (AMS) [Elaasar and Neal 2013]. RAS is an important standard proposed by the Object Management Group (OMG) to structure elements for reuse through instructions of integration and classification of artifacts in repositories and it is by definition a pivotal language for software component reuse. AMS is the core representation of the Open Services for Lifecycle Collaboration (OSLC)[1], an industrial standard aiming at provide the means to interoperate Software Engineering tools on the web through assets and services for toolchain instantiation. Thus, RAS++ is intended to support core service domain representations in MDE as a Service.

We have also created a base of assets by mining data from the ReMoDD repository [France et al. 2007], so that Software Engineers can analyze the best options to satisfy a customer request. Our intent is to use them for simulation purposes in integration sce-

[1] OSLC: `http://open-services.net/specifications/`

narios of coopetition. For example, Company A may request a DSL that complements resources used in two configurations shown on the top of Figure 1 (3). With test purposes, we represented assets in a repository for MDE Resources that serve as complement to the core assets developed previously with the FOMDA DSL. Next steps is to simulate integration with the KB we already have for business scoping MDE in Internet of Things (IoT), Web information systems and Embedded Systems.

The inclusion of the scope shown in Figure 1 (1) in MDE as a Service, requires the execution of the following opportunistic reuse steps: 1) represent and publish assets designed with the RAS++ in one or more global repositories; 2) analyze and compare resources and; 3) download and integrate the selected resources into the core assets illustrated in Figure 1 (2). These steps are assisted by our Eclipse plugin, which connects these two reuse scopes and enables Software Engineers to: 1) design, publish and download assets from repositories, and; 2) through model-to-model (M2M) transformations, transform assets from RAS++ to the FOMDA DSL. Therefore, through a pivotal language, we started to implement the overall scenario motivated in Figure 1.

Finally, in order to test implementations for Software Factory concepts, we developed seven M2M transformations that allows the transformation of RAS++ models to other integration languages.

3. Lessons Learned

Although we have progressed a little bit in this direction towards coopetition as a sustainable business model for small software houses, similar experiences are scarce and not appropriately discussed in the literature. Likewise, we can mention several other limitations hampering the implementation local and global coopetition in Software Engineering in general including:

Systematic mapping study on settings for toolchain in SE. The lack of this study hampers the execution of an appropriate comparison of our work with the state of the art. Besides, it would be benefit for future implementations of coopetition to acknowledge all the other toolboxes overlapping and complementary for automation of tasks in Software Engineering, thus characterizing an investigation on the intentions of these approaches and their properties;

Preliminary engineering phases for toolchain integration. In order to establish partnership in software process automation, it is important to reach complements for reuse steps earlier to our experiences, thus in global scales of cooperation. Finding these phases and develop appropriate toolboxes is of relevance for the execution of preliminary tool chain [Zakheim 2017], thus focused on the automation of phases of the overall software development lifecycle;

Increments for the engineering steps in the FOMDA Approach. In special, the following research topics are imperative:

- Evaluate other usefulness from a business rule system in the context of MDE as a Service [Pillat et al. 2015], such as allowing the development of block composition rules for inclusion of coopetition players;
- Application for other coopetition scenarios in MDEaaS [Basso et al. 2017a, Basso et al. 2015]. For example, conducting new investigations for other Software Engi-

neering needs than toolchain, such as complements for disruptive business models including smart contracts and blockchain as a services;

- Requirements for Software Engineering technology transfer in MDE as a Service. So far, few is understood about quality attributes associated with MDE resources, thus elements for decision making in multi Software Engineering contexts are important for business [Papatheocharous et al. 2018].

Post engineering phases for MDE resources. We look for new studies focused on the automatic introduction of MDE features through model transformations, for example, from FOMDA DSL-compliant assets to various other process modeling / execution DSLs and their reuse mechanisms [Pillat et al. 2015]. For example, many toolboxes have been adopted by software development companies, which are potential potential customers for managing co-constructed services. These add-ons could extend a possible MDEaaS market, thus opening a window into the investigation of an automatic approach to software development process automation.

Acquisition of assets. Assets and services are essential for the implementation of global coopetition in the area. In this sense, work that applies some concepts of asset acquisition should be investigated, which is promoted through asset platforms [Papatheocharous et al. 2018] and, more recently, through distributed services. So an open question is whether these new approaches are applicable to MDEaaS?

Requirements for MDEaaS. The requirements for these services are not known. We have made a small contribution in this regard, but to the purely technical elements involved in these assets. In addition, OSLC is in its infancy and there are also other approaches to toolchain that should be investigated in this context. Therefore, further studies characterizing these requirements and applying / testing existing asset specifications are welcome.

Break away from the traditional way of transferring technology to the market. Past experience [Basso et al. 2017a] has led to the understanding that small technology transfer participants have little or no chance to commercialize their assets as Component Of The Shelves (COTS). One way to break this old paradigm is by implementing MDEaaS as peer negotiations, which requires a huge effort to connect the aforementioned interdisciplinary research cooperation.

In this sense, the literature in the area does not identify the main problems that hinder this type of initiative, which could benefit small businesses. Because each company may have different contexts (technologies, team configurations, and processes), this limitation poses several risks in introducing tools and methodologies for Software Engineering [Jacobson et al. 2012]. Most importantly, we missed reports on MDEaaS scenarios, which could prevent us from pitfalls and facilitate this work by finding new opportunities and establishing partnerships.

4. Conclusion and Discussion

This paper reports the work done in MDE as a Service, an implementation of coopetition for Software Engineering contexts that adopt MDE tools for automation of some tasks on software development. To Software Engineering contexts, coopetition could benefit companies through automation in technology transfer. However, this is yet to be explored by research and practice. Hence, this is the right moment for foundation studies for automation in Software Engineering technology transfer on the large through coopetition.

This paper reports some contributions scoping local and global coopetition scenarios through two representation languages: FOMDA DSL and RAS++. FOMDA is for systematic reuse and enables the implementation of Software Factory, applied in local approaches for MDE as a Service in real settings. RAS++ is a mean to promote integration in the large in a opportunistic reuse approach. In order to promote automated technology transfer, RAS++ is a new pivotal representation language in support for services for global coopetition scenarios. In an ideal stage of research, RAS++ would allow to connect MDE resources from many providers (centralized or distributed) with support for integration in toolchain.

MDE resources are agnostic to what software houses adopt for specific applications, *i.e.*, independent from the software development process reference model adopted by a company and also independent from application domains. This means that they must be combined in theory and practice within a context characterizing coopetition blocks. Our conclusion is that, considering strictly MDE resources developed in support for MDE toolchains, the use of a toolbox that integrates assets and services makes all the sense and puts at least local coopetition as a reality in Software Engineering.

Referências

[Basso 2017] Basso, F. P. (2017). *RAS++: Representing Hybrid Reuse Assets for MDE as a Service. Av at <www.cos.ufrj.br/uploadfile/publicacao/2811.pdf>*. PhD thesis.

[Basso et al. 2006] Basso, F. P., Becker, L. B., and 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, pages 374–381.

[Basso et al. 2017a] Basso, F. P., Oliveira, T. C., Werner, C. M., and Becker, L. B. (2017a). Building the foundations for 'mde as service'. *IET Software*, 11:195–206(11).

[Basso et al. 2016a] Basso, F. P., Oliveira, T. C., Werner, C. M. L., and Frantz, R. Z. (2016a). 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.

[Basso et al. 2013a] Basso, F. P., Pillat, R. M., Oliveira, T. C., and Becker, L. B. (2013a). Supporting large scale model transformation reuse. In *12th International Conference on Generative Programming: Concepts & Experiences.*, GPCE'13, pages 169–178.

[Basso et al. 2016b] Basso, F. P., Pillat, R. M., Oliveira, T. C., Roos-Frantz, F., and Frantz, R. Z. (2016b). Automated design of multi-layered web information systems. *Journal of Systems and Software*, 117:612 – 637.

[Basso et al. 2015] Basso, F. P., Pillat, R. M., Roos-Frantz, F., and Frantz, R. Z. (2015). Combining mde and scrum on the rapid prototyping of web information systems. *International Journal of Web Engineering and Technology*, 10(3):214–244.

[Basso et al. 2017b] Basso, F. P., Werner, C. M. L., and 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*, pages 34–40.

[Basso et al. 2014] Basso, F. P., Werner, C. M. L., and Oliveira, T. C. (2014). Towards facilities to introduce solutions for mde in development environments with reusable assets. In *Int. Conf. on Information Reuse and Integration*, pages 195–202.

[Basso et al. 2013b] Basso, F. P., Werner, C. M. L., Pillat, R. M., and Oliveira, T. C. (2013b). A common representation for reuse assistants. In *13th International Conference on Software Reuse*, ICSR'13, pages 283–288.

[Basso et al. 2013c] Basso, F. P., Werner, C. M. L., Pillat, R. M., and Oliveira, T. C. (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*, pages 721–726.

[Boehm 2006] Boehm, B. (2006). A view of 20th and 21st century software engineering. In *28th International Conference on Software Engineering*, ICSE '06, pages 12–29.

[Bouncken et al. 2015] Bouncken, R. B., Gast, J., Kraus, S., and Bogers, M. (2015). Coopetition: a systematic review, synthesis, and future research directions. *Review of Managerial Science*, 9(3):577–601.

[Cuadrado et al. 2014] Cuadrado, J. S., Guerra, E., and Lara, J. D. (2014). A component model for model transformations. *IEEE Transactions on Software Engineering*, 40(11):1042–1060.

[Elaasar and Neal 2013] Elaasar, M. and 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, pages 154–169.

[France et al. 2007] France, R., Bieman, J., and 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, pages 311–317.

[Franzago et al. 2017] Franzago, M., Ruscio, D. D., Malavolta, I., and Muccini, H. (2017). Collaborative model-driven software engineering: a classification framework and a research map. *IEEE Transactions on Software Engineering*, pages 1–1.

[Fuggetta and Nitto 2014] Fuggetta, A. and Nitto, E. D. (2014). Software process. In *36th International Conference on Software Engineering*, ICSE '14, pages 1–12.

[Greenfield and Short 2003] Greenfield, J. and 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, pages 16–27.

[Hong-min et al. 2010] Hong-min, R., Jin, L., and 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, pages 1–4.

[Jacobson et al. 2012] Jacobson, I., Ng, P.-W., McMahon, P. E., Spence, I., and Lidman, S. (2012). The essence of software engineering: The semat kernel. a thinking framework in the form of an actionable kernel. *ACMQUEUE. Development*, 10(10):1–12.

[Liebel et al. 2014] Liebel, G., Marko, N., Tichy, M., Leitner, A., and Hansson, J. (2014). Assessing the state-of-practice of model-based engineering in the embedded systems domain. In *Model-Driven Engineering Languages and Systems*, pages 166–182.

[Monteiro et al. 2014] Monteiro, R., Assumpcao Pinel, R., Zimbrao, G., and Moreira de Souza, J. (2014). 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)*, pages 575–586.

[Mussbacher et al. 2014] Mussbacher, G., Amyot, D., Breu, R., Bruel, J.-M., Cheng, B. H., Collet, P., Combemale, B., France, R. B., Heldal, R., Hill, J., Kienzle, J., Schöttle, M., Steimann, F., Stikkolorum, D., and Whittle, J. (2014). The relevance of model-driven engineering thirty years from now. In *Model-Driven Engineering Languages and Systems*, pages 183–200.

[Papatheocharous et al. 2018] Papatheocharous, E., Wnuk, K., Petersen, K., Sentilles, S., Cicchetti, A., Gorschek, T., and Shah, S. M. A. (2018). The grade taxonomy for supporting decision-making of asset selection in software-intensive system development. *Information and Software Technology*, 100:1 – 17.

[Pillat et al. 2015] Pillat, R. M., Oliveira, T. C., Alencar, P. S., and Cowan, D. D. (2015). BPMNt: A BPMN extension for specifying software process tailoring. *Information and Software Technology*, 57(0):95 – 115.

[Ritala et al. 2014] Ritala, P., Golnam, A., and Wegmann, A. (2014). Coopetition-based business models: The case of amazon.com. *Industrial Marketing Management*, 43(2):236 – 249.

[Yie et al. 2012] Yie, A., Casallas, R., Deridder, D., and Wagelaar, D. (2012). Realizing model transformation chain interoperability. *Software & Systems Modeling*, 11(1):55–75.

[Zakheim 2017] Zakheim, B. (2017). How difficult can it be to integrate software development tools? the hard truth. *InfoQ*.