Applying Model-based Requirements Engineering in Three Large European Collaborative Projects: An Experience Report

Andrey Sadovykh, Dragos Truscan, Hugo Bruneliere

Applying Model-based Requirements Engineering in Three Large European Collaborative Projects: An Experience Report

Andrey Sadovykh
Innopolis University - SOFTEAM
Innopolis, Russia - Paris, France
a.sadovykh@innopolis.ru,
andrey.sadovykh@softeam.fr

Dragos Truscan
Åbo Akademi University
Turku,Finland
dragos.truscan@abo.fi

Hugo Bruneliere
IMT Atlantique, LS2N (UMR CNRS 6004)
Nantes, France
hugo.bruneliere@imt-atlantique.fr

***Abstract*—In this paper, we report on our 5-year's practical experience of designing, developing and then deploying a Model-based Requirements Engineering (MBRE) approach and language in the context of three different large European collaborative projects providing complex software solutions. Based on data collected both during projects execution and via a survey realized afterwards, we intend to show that such an approach can bring interesting benefits in terms of scalability (e.g., large number of handled requirements), heterogeneity (e.g., partners with different types of RE background), traceability (e.g. from the requirements to the software components), automation (e.g., requirement documentation generation), usefulness or usability. To illustrate our contribution, we exemplify the application of our MBRE approach and language with concrete elements coming from one of these European research projects. We also discuss further the general benefits and current limitations of using this MBRE approach and corresponding language.**

***Index Terms*—Requirements Engineering, Model-based Engineering, Collaborative Projects, Experience Report, Scalability, Heterogeneity, Traceability, Automation**

I. Introduction

In many countries, funded collaborative projects involving academic and industrial partners are a preferential way of implementing ambitious Research and Innovation Actions (RIAs) [1], [2]. Such projects are also frequently used to foster international collaborations and to develop corresponding long-term partnerships between organizations from various countries. This is notably the case in Europe, where the European Commission has several active funding bodies proposing various kinds of funding programmes targeting different societal, economic, and scientific grand challenges [3].

In the context of such collaborative projects in the Software and Systems Engineering area, possibly important in terms of the number and variety of involved partners and countries, the main expected results are generally large and complex integrated frameworks or tool sets. In order to allow for their actual design, development and deployment, it is thus fundamental to be able to support and manage as efficiently as possible the corresponding Requirements Engineering (RE) processes [4], from the initial identification of the industrial needs to their realization within the final innovative solutions.

As it can be observed in the literature, the state-of-art in RE is already rich in terms of approaches and corresponding technical solutions [5]. Among other paradigms, some of these approaches rely on model-based concepts [6]. This notably reflects the fact that model-based principles and techniques have become more popular in industry over the last two decades, as they can provide relevant abstraction, genericity or reusability capabilities (for example) [7].

In this paper, we report on our practical experience of proposing and applying a Model-based Requirements Engineering (MBRE) approach and language during 5 years in the context of three different large European collaborative projects, each one of them providing as a result various complex software solutions (e.g., frameworks, integrated tool sets, etc.). The approach was mainly developed by the SOFTEAM company and applied in 3 different research projects in which SOFTEAM participated in the technical coordination activities. With this reporting and data analysis work, we notably intend to show that such a MBRE approach can bring interesting benefits in terms of scalability (e.g., large number of handled requirements), heterogeneity (e.g., partners with different profiles and types of RE background), traceability (e.g., from the initial requirements to the software components), automation (e.g., requirement documentation generation), as well as general usefulness or usability.

The remainder of the paper is structured as follows. Section II introduces the general context and background of this experience report. Then, Section III highlights relevant related work while Section IV describes the MBRE approach we propose in order to support RE processes in collaborative projects. Section V evaluates this approach and its practical application in the context of three different large European collaborative projects. Finally, Section VI discusses the main lessons learned from this experience while Section VII concludes the paper.

This work has received funding from the ECSEL Joint Undertaking under grant agreements No. 737494 (MegaM@Rt2 project) and No. 101007350 (AIDOaRt project), from ITEA3 (REVaMP2 project No. 15010), and from H2020 under grant agreements No. 732064 (DataBio project) and No. 957212 (VeriDevOps project).

II. Context and Background

European collaborative projects are one of the main sources of innovation in Europe [2]. There are several funding bodies such as ECSEL, Horizon 2020, ITEA, etc. coming with different types of funding programmes. The funded projects all have in common that a number of organizations (academic or industrial) from several European countries participate in a collaborative research effort. The average number of partners in such projects varies according to the funding framework. For example, this goes from 4.69 in Horizon 2020 [8] to 30 and 40 for Research and Innovation Actions and Innovation Actions ECSEL projects respectively [9]. However, it is not unusual that ECSEL projects exceed 100 organizations [10].

A key element of such European research projects is the complementarity of the project's participants. These participants can come from different application domains (e.g., railway, avionics, telecom, manufacturing), have different sizes (e.g., from small and medium enterprises to large industrial groups), levels of maturity or come with different kinds of research background. They have to work together to achieve a set of shared R&D goals, usually validated via several case studies that serve as a common platform for experimenting on newly designed and developed technologies. The objective is to provide evidence to the European Commission of the benefits and drawbacks, both scientific and economical, that the developed innovative technologies can offer.

However, the diversity and number of partners can also imply project management challenges related to 1) the elicitation of the needs from the industrial case study providers and 2) the identification of not only the concrete solutions to be provided during the project (lasting typically three years), but also of a roadmap for the development of the final technical solution. These challenges are amplified by the number of partners in the project and by the diversity of their (scientific and technical) backgrounds and of their application domains [11]. When such challenges are not addressed properly, they can negatively influence the outcomes of the project [12].

Therefore, appropriate RE practices are necessary for maximizing benefits and for achieving good technical results in such projects. Those RE practices span over all areas of the RE process such as requirements elicitation, specification, validation and management [4]. The approach proposed in this paper applies model-based principles and techniques to support and improve the RE process in possibly large and diverse collaborative projects. With our approach, we intended to provide better scalability, improved support for heterogeneous types of partners, and more complete tool support / automation for managing the requirements of the developed solutions.

In practice, the proposed approach was designed, developed, matured and applied in three large European projects covering a period of five years (cf. Section V-A for more details): H2020 **DataBio**[1] 2017-2019 [13], ITEA3 **REVaMP2**[2] 2016-2019 [14], ECSEL **MegaM@Rt2**[3] 2017-2020 [15].

In this paper, we specifically focus on the three above-mentioned projects as the most recent ones from our side. However, we have already faced the similar challenge to coordinate RE processes and tool framework developments in more than 15 similar European collaborative research projects.

III. Related Work

By nature, RE always implies a modeling activity [16]. Thus, model-based principles and techniques have already been applied to address different kinds of RE activities [6]. Notably, this resulted in acknowledged contributions such as goal modeling languages for instance [17], [18] or the ReqIF requirement modeling standard [19]. However, up to our current knowledge, few model-based approaches have been proposed to cover complete RE processes in the general case. For example, an existing solution relies on a generic modeling framework to represent and simulate requirements independently from the context [20]. Another one is based on a core requirements metamodel to be customized to support various kinds of RE processes [21]. In addition, we can also mention transformation-based approaches from goals models to design models, such as the KAOS method [22].

However, these approaches have been deployed only in a single project [23] or were mostly focusing on some particular RE aspects, such as requirements visualization for instance [24]. In this paper, we rather intend to propose a model-based RE approach to possibly cover complete RE processes in large collaborative projects with heterogeneous and numerous partners, including important features such as requirements traceability and automated documentation generation. From a more industrial perspective, and to foster genericity, reusability and interoperability, the proposed MBRE approach and language are rooted in acknowledged software and system modeling standards: UML [25] and SysML [26]. Moreover, they are also partially inspired by the European Space Agency standard terminology and structure for RE documents [27].

Closer to the context and background of our work, a few model-based approaches have already been proposed and used inside collaborative research projects to handle, at least partially, corresponding RE processes. For instance, Nielsen et al. [28] proposed a RE process for small EU-funded projects. They proposed to have each case study provider pairing with an academic partner to assist the former in eliciting and specifying the requirements. Differently from this approach, we propose to use model-based techniques to elicit generic framework requirements as a meeting point between the case study requirements and the tools developed in the project. We also enforce traceability between requirements and use the requirements specifications as the reference for performing gap analysis, road mapping and generating documents.

In a different work [11], the authors discuss lessons learned in a large scale research project and exemplify different techniques for elicitation, traceability and gap analysis. They

[1] https://www.databio.eu/

[2] http://www.revamp2-project.eu/

[3] https://megamart2-ecsel.eu/

partially use model-based techniques for elicitation and analysis or to identify the framework architecture, but did not not applied at the entire approach level. They also defined metrics-based validation criteria for requirements that we could integrate in our approach in the future.

Another work focuses on identifying the framework architecture in an European project [29]. However, the approach is not model-based and the links between the architecture and the tools provided by technology providers are missing, thus making gap analysis more difficult to perform.

IV. Proposed Approach

As a solution to the problem of covering complete RE processes in the context of large collaborative projects involving many different kinds of partners, we propose a MBRE approach with a dedicated modeling language and the corresponding tool support. We give an overview of this approach in Section IV-A and we describe the corresponding RE language in Section IV-B. In Section IV-C, we also provide more details on how these approach and language have been concretely implemented and supported in the Modelio tool.

A. Overview of our Conceptual Approach

Our MBRE approach is defined to take into account the typical participant roles and activities in large collaborative research projects involving both academic and industrial partners. From our long-term experience in many projects of this type, both at the national level in each of our respective countries and at the international level (e.g. in Europe), we have been able to observe a common general organization: *Case Study Providers* have a practical problem, e.g., in terms of development processes, product quality or features, for which they are looking for innovative solutions; *Research Partners* develop new research methods and prototypes and *Technology Providers* offer technological solutions, in collaboration with the research partners, that can be deployed and evaluated against the industrial case studies. It is then the task of a *Technical Coordination Team* to ensure a smooth collaboration between the involved partners and that the project's objectives are globally achieved. Of course, there are different kinds of project and the participants role may slightly vary: A same entity can sometimes be both a Research Partner and a Technology Provider, or a Case Study Provider can also become an actual customer of a Technology Provider. However, we believe the proposed global organization is adapted to a majority of the large collaborative research projects we target.

As illustrated in Figure 1, our approach proposes that the technological solutions are provided in the form of a generic **framework**. The framework requirements are elicited from the industrial case study requirements. The framework aggregates **Tool Components** specific to different application domains and applicable to one or several case studies. Each tool component is developed by different project participants based on **Tool Component Requirements**, also sometimes called "Tool Purposes", which satisfy the framework requirements. Each tool component will have a specific architecture including the interfaces available for being interconnected with other tools. This notably allows for different tool chains to be easily created for particular case studies. Each tool component is developed with different priorities and becomes available at different milestones from the **Tool Component Roadmap**.

Based on the individual tool component architectures and roadmaps, the technical coordination team can design the **Framework Architecture** and the **Framework Development Roadmap**. Such a roadmap will allow the case study provider to know when different technologies will be available for evaluation, and consequently to create **Case Study Requirements Validation Roadmaps**. When different **Tool Component Implementations** become available at different milestones of the project, they are integrated in the framework and evaluated against the case studies in the **Case Study Requirements Validation and Solution Evaluation** process.

The above process is applied continuously during the implementation of the project. However, several challenges have to be addressed for ensuring a smooth process.

The first challenge is the elicitation of the framework requirements that support the needs of the industrial case study providers (i.e., the case study requirements). This is not trivial since it requires the collaboration of all the partners in the project, each one coming from different application domains and having different technical backgrounds. Having a centralized and clear mapping between case study requirements on one hand, and the framework and tool components requirements on the other hand, also allows the technical coordination team to track the progress, to spot further needs for technical solutions and to mitigate the risks.

The second challenge that we address is to create a roadmap for the development of the framework by collecting development plans for individual tool components. This will allow all partners in the project to be aware of when different features of the framework will be implemented. This will also allow case study providers to know when these tools can be evaluated against their own case studies. In addition, having such a roadmap allows the technical coordination team to better plan and produce different deliverables, demonstrations and thematic events in the project.

B. A Dedicated Modeling Language for RE

To realize the proposed conceptual approach, and to represent and share appropriately the requirement data during the full RE process, we worked on a dedicated modeling language for MBRE. The reason for developing a dedicated language, rather than directly using an existing one, is that the commonly used modeling languages (i.e. general-purpose ones such as UML) are very wide in terms of scope. Thus, users tend to have different ways to specify requirements and corresponding design decisions when using them. This is actually an issue in our context of large collaborative projects involving partners with different backgrounds and experiences.

As a solution, we decided to design and build our modeling language by following a bottom-up approach: We started by analyzing what were the documents (i.e. deliverables)

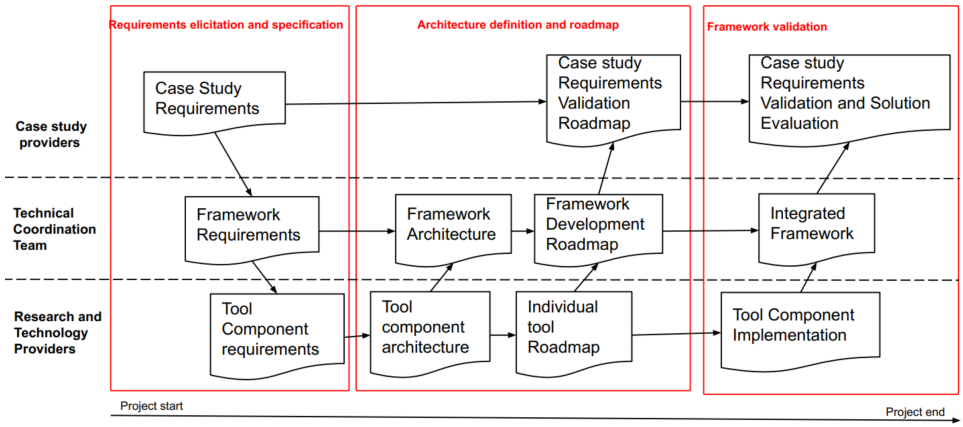

Fig. 1. A conceptual approach for MBRE.

generally needed in terms of both requirements and architecture in large collaborative projects. To this end, we notably studied standard representation formats such as ESA ESS [27] and standard modeling languages such as UML [25]. Then, inspired by them, we designed a generic modeling language that would help in supporting and simplifying the automated generation of these documents/deliverables. This language was designed based on the needs of the requirements engineering processes used in the three European projects mentioned above. But it can be customized and extended if needed according to the processes of other projects. The objective of the language is to allow the project participants to elicit, specify and then share both the requirements and corresponding architectural elements which are to be addressed by following the proposed conceptual approach (cf. Section IV-A). This is why we made the effort of using a common terminology between the conceptual approach and the RE modeling language.

1) Abstract Syntax: The main concepts of our dedicated modeling language for RE are depicted in Figure 2.

At the Requirements level, the base element is the **Requirements Container** that is used to logically group sets of related requirements (e.g., requirements attached to a same tool). Each **Requirement** is represented by various properties such as an *identifier*, a *definition*, a *criticality* level, a corresponding *release*, a *status* and additional *comments* if any. The *criticality* level indicates the priority for the requirement's implementation (i.e. low, medium or high). The *release* indicates the milestone at which the requirement is planned to be satisfied (i.e., baseline, initial, intermediate or final. If needed, the possible releases can be modified and configured in the beginning of the project based on its own actual milestones. The *status* indicates the state of fulfillment of the concerned requirement (i.e. planned, in progress, done, postponed or cancelled). The *comments* provide additional information on the concerned requirement. A given requirement can be connected to another requirement it depends on via a *trace* dependency link. For example, a **Tool Requirement** (also called Tool Purpose) can be linked to a corresponding **Framework Requirement**, and this Framework Requirement to a corresponding **Case Study Requirement**. Moreover, any architectural element realizing a given Requirement can be mapped to it via a *satisfy* dependency link. We can keep track of the source of the requirements using *RequirementsContainers* and *ID* naming conventions for the project participants.

At the Architecture level, the base element is the **Package** that is used to logically group sets of related **Framework or Tool Components**, common interfaces of platform nodes. Each **Component** depicts a tool or its constituent part. A given component can be composed of different *sub-components* to represent its various constituent parts. An **Interface** describes a required functional service (e.g., XMI import/export). It can be connected to a component via an interface realization link to indicate a *provided service*, e.g., the tool component's output, or via an use dependency link to indicate a *consumed service*, e.g., the tool component's input. A **Node** represents a deployment platform (e.g., Eclipse RCP, a Java virtual machine, etc.). A given component can be connected to different nodes, i.e. its *deployment platforms*. It is important to note that the metamodel is voluntarily not defining any specific element to support the framework validation phase (cf. Figure 1). This is mainly because that phase is a manual activity where different case study providers instantiate tool chains from the framework by taking advantage of the traceability between requirements, and run them on different validation scenarios. However, elements for requirements validation metrics could be added in the future to be used both at case study and project level in connection with the project objectives.

2) Concrete Syntax: Attached to the abstract syntax of our language as introduced just before, various kinds of concrete notations could be envisioned, either graphical, textual or combining both. Based on our own experiences, and notably the industrial experience of SOFTEAM when using Modelio-based solutions for their customers' projects, we decided to consider a combination of tabular views and UML/SysML-like diagrams as the syntax for our RE language.

In terms of diagrams, a SysML Requirements diagram is used to graphically map the different tool components to their respective tool requirements and then to their corresponding

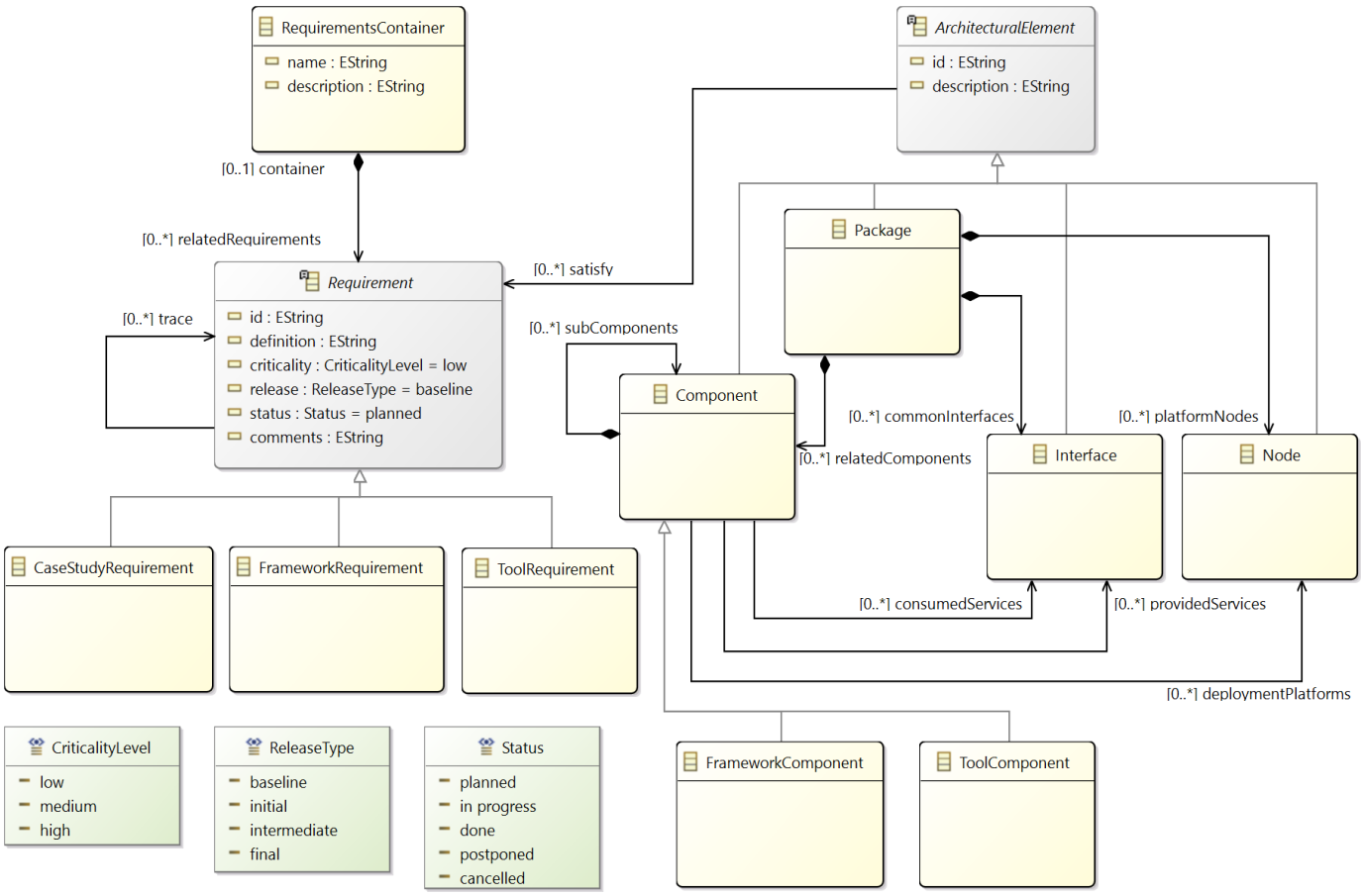

Fig. 2. Metamodel describing the abstract syntax of our RE modeling language.

case study requirements. A UML Class diagram is used to both describe a component, their constituent parts and its required and provided interfaces. It is also used to map individual tool components to conceptual frameworks components. A UML Deployment diagram is used to represent the nodes showing the deployment constraints of the different tool components.

As far as tabular views are concerned, they are mostly used in order to easily enter and then properly display various properties of possibly very large sets of requirements.

Figure 3 illustrates this concrete syntax, as currently implemented in the Modelio tool (cf. Section IV-C), on an example RE model from the MegaM@Rt2 project. In this particular example, we can see in the tabular view a set of tool requirements/purposes for the MATERA2 tool (upper-right view). These tool requirements are grouped in a same requirements container named “MATERA2 (ABO)” (cf. left panel). On the related diagram (cf. bottom view), we can see how a given Modelio tool requirement “MATERA2-010” is traced to some framework requirements “SYS-010100”, etc. and then, by transitivity, to some case study requirements “IKER_01”, etc. We can also see that a given tool component “MATERA2 (ABO)” satisfies a corresponding tool requirement “MATERA2-010”. To give a example, “IKER_01” case study requirement stands for ”A modelling tool supporting DDS UML profile, for the representation of DDS related concepts in order to enable code generation based on it”. By analysing this and similar requirements the following framework requirement may be derived: ”SYS-010100: The SE must support standard modelling languages, standard profiles (i.e. AADL, UML, SysML, MARTE, fUML, UTP) and profile customisation capability”. ABO, the tool provider for MATERA2 tool, further interpreted these requirements and mapped them to the purposes/requirements of their tool in the following way: ”MATERA2-010: MATERA2 shall provide UML based modeling and executable specifications”.

3) Semantics: As partially extending UML core concepts, the semantics of our RE language is directly connected to the semantics of UML. For instance, we consider the concepts of **Package**, **Component**, **Interface**, **Node** and various types of links as having the same meaning as prescribed by UML. However, in order to make the models universally understandable and reusable by other persons relying on our RE language, we limited their usage to the strict definitions provided in Section IV-B1. As introduced earlier, one of the main objectives of our RE language is to allow for the automated generation of corresponding requirements, architecture and roadmap documents. Thus, RE models expressed in our language are meant to be interpreted by (i.e. taken as inputs

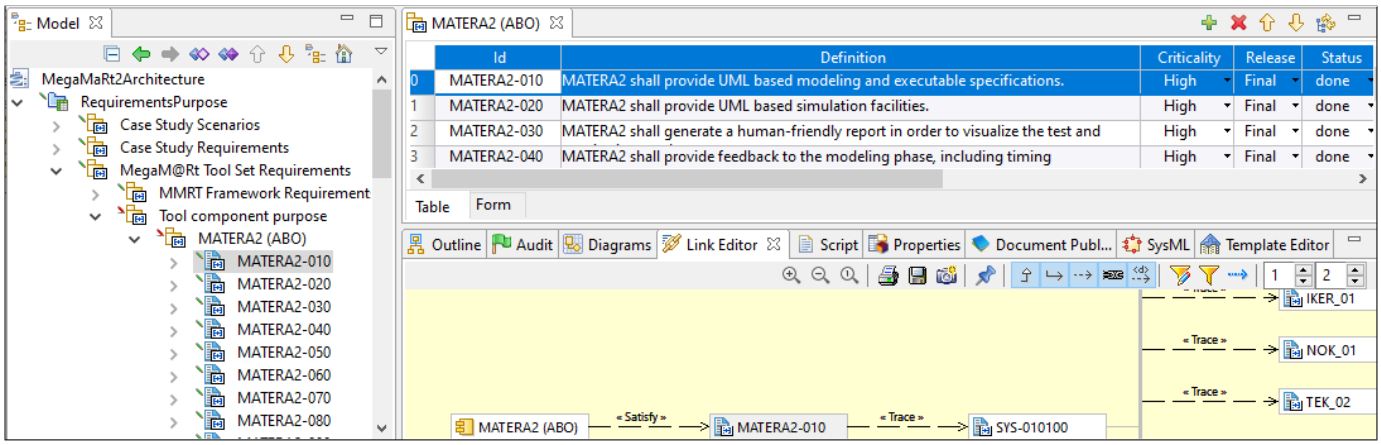

Fig. 3. Example view of the concrete syntax of our RE modeling language, as implemented in the Modelio tool.

of) specific document generators. As a consequence, part of the semantics of our RE language is also embedded in the source code of these document generators. For example, a **Package** would often correspond to a first-level chapter in the specifications. A *Common Interface* is a section of the Software Requirements Specification (SRS) where we list the information that is very helpful for integration. Besides, each **Tool Component** section will have exactly four subsections based on the used diagrams (cf. Section IV-B2): The SysML Requirements diagram is the main source to extract information about the features (existing or planned) or a component, the UML Class diagram maps to the corresponding section giving more information on provided and required services, etc. Basically, the document generator would navigate the whole model from its root and, when finding an element with a corresponding type e.g., a Package providing Common Interfaces, it will collect the corresponding data from the model and integrate it into paragraphs, tables, matrices and figures of the generated document, cf. a deliverable generated in MegaM@Rt2 for example [30].

C. Implementation of the RE Approach in Modelio

The MBRE approach and language presented in this paper could be possibly implemented by relying on different modeling tools (either open source or proprietary). However, we made the choice of implement them in the Modelio tool [31] because it is developed by SOFTEAM that was a core partner in each of the three projects we report on in this paper. This notably allowed us to benefit from an extensive support during the design, development and deployment of the solution.

In the DataBio project, we first experimented with extending/refining ArchiMate as the high-level representation for architecture. However, due to the constraints imposed by the ArchiMate metamodel, collaborative modeling was quite tedious in Modelio. For example, the locking of a single element was leading to the locking of a large portion of the model, possibly impacting other users. Moreover, ArchiMate is quite new and much less adopted in the industry. This would have resulted in a steeper learning curve, such a limitation being partially reflected in the survey data presented in Section V-B. These are the reasons why, in the subsequent REVAMP and MegaM@Rt2 projects, we rather opted for extending/refining UML as 1) allowing atomic editing of components and 2) being already widely known in the industry.

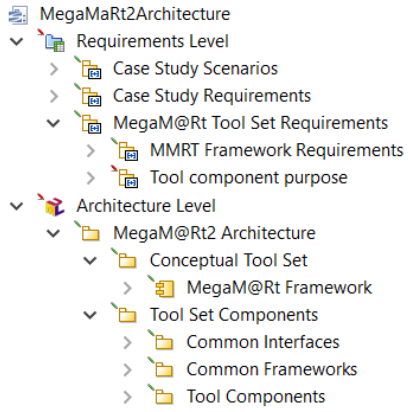

Fig. 4. Structure of the RE model in Modelio: Example from MegaM@Rt2.

The main elements of a RE model in Modelio are shown in Figure 4, with respect to the MegaM@Rt2 project as a practical example. Conforming to our RE language as described in IV-B, the overall structure of the RE model includes both the Requirements and Architecture levels. For the Requirements level, we directly benefited from Modelio Analyst features such as Requirements Containers, Tabular, Diagram view, Traceability matrices and Import from Excel. Modelio also allowed us to specialize the Requirements properties (that correspond to the columns in the tabular view) in order to implement our approach and language. For the Architecture level, we relied on the standard implementation of UML as currently available in Modelio. We refined this UML implementation in order to limit the use of UML to the concepts considered in our RE language. Moreover, in order to facilitate the initial use of the RE language, we also built-in a specific template for tool components and made it

available directly from the Modelio workbench. This way, the users of our RE language in Modelio can benefit from clear guidelines on what is expected from them when elaborating on their Requirements model.

The Modelio Document Publisher was also a major feature to construct our MS Word document generator. Indeed, Modelio provides a simplified document template editor with graphical interface that has pre-build functionalities for navigating the model, filtering model elements, extracting textual notes and diagrams, as well as building sections, paragraphs, tables and matrices. These features were particularly useful to generate important parts of our documents. This is notably the case for roadmaps, where we created tables displaying mappings between case study requirements and tool requirements indicating planned delivery dates. Based on such roadmaps, the case study providers could anticipate and organize both the building of their tool chains and their validation activities. Moreover, the traceability information in the RE model was extremely important to identify the case study requirements that are not addressed by any of the tool components. This way, we were able to conduct gap analysis from our RE model in order to better plan corrective actions.

In addition to the base implementation of our MBRE approach and corresponding language, our Modelio-based implementation also provides additional features which are particularly relevant in our context. For example, it is possible to relate and trace our RE models to elements coming from other kinds of models in Modelio (Enterprise Architecture models expressed in ArchiMate [32], business models in BPMN [33], system models in SysML [26], etc.). It is also possible to import requirements from external tools (e.g., Microsoft Excel) and to integrate them in our RE model. Moreover, new generation capabilities can potentially be added to target other kinds of text-, diagram-, table- and matrix-based representations for the RE information. Finally, our Modelio-based implementation directly benefits from the collaborative features provided by the Modelio environment: support for (un)locking model elements, commit and update, configuration or version management, etc.

V. Evaluation

To evaluate our MBRE approach and language, we collected various kinds of data associated with the three large collaborative projects we considered. We describe quantitative and qualitative data resulting from both the projects' execution, in Section V-A, and a survey realized after the end of these projects, in Section V-B. In Section V-C, the collected data is then used to assess the relevance of our RE approach and language according to main targeted properties.

A. General Data on Projects

The proposed MBRE approach and language have been designed, developed and then deployed in the context of three large collaborative projects, financially supported by the European Commission under different R&D funding programs.

The first project is named **DataBio**, standing for Data-Driven Bioeconomy (Horizon 2020). It lasted 3 years from 2017 to 2019 and involved 48 partners from 17 countries, for a total budget of € 15M. 27 case studies in the agriculture, forestry and fishery areas were considered during this project.

The second project is named **REVaMP2**, standing for Round-trip Engineering and Variability Management Platform and Process (EUREKA ITEA3). It lasted 3 and a half years from 2016 to 2019 and involved 27 partners from 5 countries, for a total budget of € 22M. Seven industrial use cases in the cyber-physical systems, electronic systems or tourism areas were considered during the REVaMP2 project.

The third and last project is named **MegaM@Rt2**, standing for MegaModeling at RunTime - An scalable model-based framework for continuous development and runtime validation of complex systems (ECSEL). It lasted 3 years from 2017 to 2020 and involved 27 partners from 6 countries, for a total budget of € 16.7M. Nine industrial use cases in the aeronautics, warehousing, automotive, construction, transportation or telecommunication areas were considered.

Complementary to these global projects figures, we provide in Table I additional data displaying the level of activity in terms of RE, as registered during these three projects.

TABLE I
Key figures related to the DataBio, REVaMP2 and MegaM@Rt2 projects in terms of RE activities.

Number of ...	DataBio	REVaMP2	MegaM@Rt2
Registered users	55	43	56
Contributors	31	24	27
Commits	958	534	1322
Handled requirements:	=188	=535	=428
(Case Study r.	77	190	106
+ Framework r.	104	56	91
+ Tool r.)	NA [5]	289	231
Model elements[6]	=5406	=3307	=4742
(Requirements level	535	1091	2351
+ Architecture level)	4871	2216	2393
Pages generated	61	109	125

[5]In DataBio, there was no clear separation of framework and tool reqs.
[6]Includes all applied elements e.g. attributes, relations and diagrams.

The data show that the number of individuals involved in the RE process, whether they are just registered users having a "read-only" profile or actually active contributors having a read-and-write profile, was relatively important and globally similar in the three projects.

In terms of RE activities, as illustrated in our case by the number of commits on the Requirements model, we observed a disparity between the three projects even though it stays globally important in all of them. This can be explained by the slightly different size of the three projects, e.g., number of partners or use cases. This can also be partly explained by the nature of the single commits: A given user can frequently do small single commits while another can rather commit a large number of updates as a single commit.

In terms of the Requirements models themselves, the data highlight the globally high number of handled requirements and related elements in the context of the three projects.

We can observe differences between the projects but, as stated before, this can be explained by the specificity of each particular project e.g., the number of partners and the use cases to be covered. These differences are also directly reflected and visible in the number/size of the various project's documents or deliverables generated from the Requirements models in the three projects. We also want to note that the initial use of ArchiMate in the DataBio project, as previously mentioned in Section IV-C, does not have a significant influence since the number of concepts considered for modeling the architecture is exactly the same than in UML. The slightly bigger numbers can be rather explained by the greater size of the DataBio project in terms of involved partners and tools.

We can observe that there is a significant difference between the number of requirements level elements and architecture level elements in the three projects. There are several possible explanations: 1) Architecture specification requires many elements, e.g. interfaces, components, relations, in order to satisfy a set of requirements, 2) the UML metamodel usually requires the creation of several elements for a given goal, e.g. association ends are created when an association is added (and these are also counted).

B. Survey for Projects' Participants

In order to be able to evaluate our MBRE approach and language according to more data, we have also run a complementary online survey among the members of the MegaM@Rt2, REVAMP and DataBio projects. The key research hypothesis we want to validate is whether our approach is relevant and helpful in the context of large collaborative research projects.

In the following, we summarize the main findings of the survey, while the complete survey results are available at [34].

We started with three quantitative assessment questions:

- *Q1: In your opinion, did you find this graphical model-based approach useful in different activities of Requirements Engineering? Followed by the list of main RE activities.*
- *Q2: In your opinion, do you see the modeling approach as an improvement compared to other non-modelling (e.g., text-only or table-based) regarding the following aspects? Followed by the list of characteristics that the requirements have to follow, such as correctness and traceability [35].*
- *Q3: In your opinion, did you find the following Modelio tool features useful in different Requirements Engineering activities? We listed all presumably key features of Modelio that could be considered helpful.*

In addition to these quantitative assessment questions, we proposed to answer three qualitative assessment questions:

- *Q4: In your opinion, what was the most challenging aspect of the Modelio-based approach?*
- *Q5: In your opinion, what was the most useful aspect of the Modelio-based approach?*
- *Q6: In your opinion, which additional Modelio tool features would have been useful for RE in the project?*

The potential respondents were all project partners who had an account in the shared model repository and presumably had access to the modelling. We excluded the authors from the survey in order to avoid the subjectivity bias, even though they were primary beneficiaries of the approach as responsible for architecture definition in the MegaM@Rt2 project.

In total, we had 154 individual contact persons: 55 for DataBio, 43 for REVAMP and 56 for MegaM@Rt2. We received 15 complete answers, including 1 person not involved in RE activities. We explain this level of participation by a couple of factors: (1) Few of these contact persons were active contributors to the RE process in these projects, most of them were "readers" or contributed very few elements; (2) It has been at least 1 year since the projects terminated and at least 2 year since the end of the corresponding RE work. While this amount of data cannot be considered statistically representative, we believe the received feedback still provides interesting and relevant complementary insights.

The majority of the respondents considered that our model-based RE approach was useful for different RE activities. On average, 90% in average would agree that the proposed graphical model-based approach is useful for RE (Q1), 65% would agree that the modeling approach is better for RE (Q2), and 79.59% would agree that the implementation of the approach in Modelio was useful for RE (Q3).

In Q1, all respondents (100%) agree that model-based approaches are useful in Requirements analysis and negotiation. However, only 80% find that model-based approaches are useful in Requirements validation.

In Q2, the highest agreement (86.67%) concerns the advantages of a model-based approach regarding Traceability: It can be linked to system requirements, designs, code, and tests. The lowest agreement (53.33%) concerns the benefits of model-based versus non-model-based approaches for dealing with Correctness (it accurately states a customer or external need) and Clearness (it has only one possible meaning).

In Q3, all respondents (100%) would find Changing/adding dependencies useful in the implementation of our approach. The lowest agreement (57.14%) concerns the usefulness of Roadmapping (e.g., setting up expected delivery dates and completion stage for designed components). However, this feature was introduced in the Mega@Rt project only, where 80% (4/5) of the respondents would find it useful.

Moreover, additional questions on the appropriateness of the approach resulted in the following assessments:

- 85.71% would find the approach appropriate for the given size and scope of the project.
- 92.86% would find the tool support useful for guiding the RE process and enforcing project conventions
- 71.43% would find the approach easy to learn.
- 78.57% would find the approach easy to apply and would use it again in the future.

Finally, we also received some open qualitative feedback. Respondents generally mentioned the difficulty to convince stakeholders to apply the approach as well as difficulties in terms of model synchronization during collaborative editing. More positively, respondents also mention their appreciation in terms of the standardization of the RE process provided by our approach, the improved traceability support or the fast

and clean documentation generation. However, the respondents also proposed several improvements according to their past and present practices. Notably, they suggested to support in the future a better connection with other tools, a more advanced traceability between our RE language and other languages (e.g. for software design and development), and to provide more structured templates for the requirements description.

C. Overall Assessment of our MBRE Approach and Language

Based on all the collected data presented in Section V-A and Section V-B), we can provide a first assessment of the relevance of our MBRE approach and language according to the main properties introduced in Section I:

• **Scalability** - The consolidated data extracted from the Requirements model repository in the three projects clearly shows that we have been able to support a significant number of users, including regularly active ones, as well as to handle a relatively large number of requirements and related model elements. Moreover, the final success of the three projects in terms of deliveries (i.e. documents, tools, demonstrators) and the results of the survey also demonstrate the general scalability of our MBRE approach and language, at least for projects going up to the size of the three mentioned ones.

• **Heterogeneity** - The varied characteristics of the three projects (e.g., different partners providing different kinds of tools and technologies, various use cases covering several application domains) show that we have been able to handle a certain level of heterogeneity. Based on our experience in these three projects, and the participants' feedback we collected via the survey, we can also argue that our MBRE approach can be applied similarly in any large collaborating project whose main purpose is to produce integrated software solutions.

• **Traceability** - Looking at the results of the projects participants survey notably, we can state that traceability and the capability to guide and enforce full RE processes from the beginning to the end of collaborative projects has been one of the most acknowledged features of our MBRE approach and language. Even though improvements and/or extensions are still possible, e.g., tooling support for traceability down to the source code, the feedback received up to now already highlights a correct support for traceability.

• **Automation** - The quantitative data collected at the end of the three projects show that we have been able to automatically generate documents of significant sizes from the corresponding Requirements models. These were quite complete requirements, architecture or roadmap documents whose content could be then reused directly to produce the official projects deliverables. The achieved level of automation, though not total, was already quite appreciated by the projects' participants.

• **Usefulness and Usability** - The fact that our MBRE approach and language have been successfully used in practice in the context of three different collaborative projects already shows a certain level of usability. The qualitative data collected from the participants' survey also confirm that the users globally found our approach useful during these three collaborative projects. Furthermore, they appreciated in particular the fact that the approach was easily applicable in their respective contexts, i.e. both easy to learn and to apply.

VI. Discussion

Our evaluation results are globally similar to the results observed in the context of the other approaches we discussed in the Section III. This is notably the case with respect to gap analysis, roadmapping and traceability for requirements. In what follows, we discuss further the contributions proposed in this paper as well as our own experience in their contexts. We notably present general lessons learned from the three collaborative projects in Section VI-A. We also describe in Section VI-B a few threats to validity we have identified concerning the current evaluation of our MBRE approach and language. Finally, we introduce in Section VI-C some open challenges concerning both our approach and its applications in other contexts in the future.

A. Lessons Learned

From our global experience of designing and then applying our MBRE approach and language in the context of three different large collaborative European projects, we have been able to extract some general lessons learned we hope to be useful to the RE community as well as more globally to the whole Software Engineering community:

• **Project management** - The received feedback shows that our MBRE approach and language is mostly beneficial from a project management perspective: One of the most valuable user features is the automation capability. Notably, the possibility to perform more easily gap analysis and obtain a corresponding roadmap, or to generate long documents directly from the Requirements model, were highly appreciated.

• **Framework architecture** - The proposed approach and language was relevant for defining the framework architecture. Notably, the mostly appreciated features were the support for collaborative work, the integration with a model repository, etc. The combination of well-known diagrams coming from UML and SysML with tabular views was also perceived as a good way to facilitate the use of our solution, e.g. for partners already having experiences in modeling activities.

• **Learning curve** - Some participants in our projects had a somehow limited experience in modeling, both conceptually and technically. This resulted in difficulties for them to catch up with some of the concepts in our RE language. It also appears that the guidelines initially provided, on the approach/language and Modelio tooling, were not sufficiently detailed to allow for a first usage. This was quickly fixed by the Modelio team via online hands-on sessions for example. The participants were then more easily able to go on with their RE activities in the context of their respective projects.

B. Threats to Validity

The main threat to validity concerned the amount of data we have been able to gather. Our MBRE approach and language have been deployed in "only" three different projects from which we have extracted mostly quantitative data. However,

the fact that these were large projects that run over 5 years in total already provides a certain level of confidence about the quality and relevance of the collected data. Moreover, we complemented this quantitative data with extra data (quantitative but also qualitative this time) collected from a survey largely distributed among the three projects participants. Obviously, it would have been more significant to get more answers to the survey. However, we believe the collected feedback coming from 15 different participants is already interesting in order to improve our global appreciation of the proposed solution. Another threat stands in the fact that the the evaluation of the proposed approach included the modeling language, the process and the tool. We have tried to reduce this bias by having separate groups a question for each of them in the survey, still the separation was not completely achieved.

C. Challenges

As a result of the work and experience reported in this paper, we identified some challenges we believe to be worth investigating as far as MBRE or more generally RE is concerned:

- **From requirements to source code** - Our requirements models contain information concerning mostly the needs and architectural decisions at the project level. While this is already useful for coordinating the common global effort towards the realization of the target solution, it remains relatively far from being full model-driven development where implementation and verification artifacts can be produced from the models. Thus, more efforts still have to be made in order to better fill this gap between the architecture and development levels in our MBRE solution and also possibly in others.
- **User training and support** - In our RE language, we deliberately restricted the UML usage to a limited number of concepts. We also provided tooling support, user guidelines and online workshops to make the life of the various projects' partners easier. Nevertheless, the partners appeared to still need constant support with the tooling and the approach. Thus, we strongly believe that the usability and learning curve of RE solutions are key elements to consider and improve accordingly in order to allow for their large industrial dissemination.
- **Collaborative and online work** - One of the most reported issues concerned restrictions in the collaborative editing capabilities provided by our MBRE solution as relying on a Subversion-based lock - edit - commit - release operation mode. The fact is that, nowadays, users generally tend to prefer online editing collaboration modes e.g., via their favorite Web browser. However, we have seen limited support for that so far in existing modelling tools or even in popular IDEs. Such a support for online collaboration at the requirements level is probably a path to be explored more deeply in the future as far as RE solutions are concerned.
- **Automation and production** - There are still open challenges related to the support for automation in RE processes and more generally in Software Engineering processes. For instance, we could have considered to build-in some more automated support for document generation or even code generation. However, it is always a matter of cost/benefit balance since development resources can be limited in collaborative R&D projects. Thus, our MBRE solution is still not in full production stage e.g., it requires some level of customization for each new project, and it will need more work to be made available as an actual product in the future.

D. Next steps

The very next step of our work will consist in improving our experience with the proposed MBRE approach and language by applying in the context of another large collaborative project. This will actually be the case as we are now starting RE activities in the context of two new European projects - VeriDevOps (H2020) [36] and AIDOaRt (H2020 ECSEL) that will run 3 more years and involve more than 40 partners, both industrial and academic ones, coming from 7 different countries. As we (the authors) are strongly involved in these new projects, we have already planned to work on deploying again our MBRE approach and language. This way, we hope to be able to 1) collect more relevant feedback from our partners and 2) upgrade our solution accordingly. For instance, similarly to what we did in the MegaM@Rt2 project, we plan to experiment further on the use of our MBRE approach and language in the context of hackathons mixing both business and technical people [37]. Moreover, we plan to run several controlled experiments with the objective to evaluate and compare our approach against other RE approaches.

VII. Conclusion

In this paper, we reported on our practical experience of proposing and deploying a Model-based Requirements Engineering approach and language during 5 years in the context of three different large European collaborative projects providing complex software solutions. Our MBRE approach and language mostly focused on supporting three complementary aspects: 1) requirements are described appropriately at different abstraction layers (case study, framework, tool), 2) requirements can be better interconnected and traced during the RE process and 3) requirements can be used to (semi-)-automatically perform gap analysis, roadmap and corresponding document generation.

Based on this global experience and the collected data, we showed that our MBRE approach and language can bring interesting benefits in terms of scalability, heterogeneity, traceability, automation, general usefulness or usability. We also discussed the added-value of our solution from a project management and architecture perspective while identifying some limitations we faced, in terms of RE solution learning curve for instance, and that we already managed to partially overcome. As a conclusion, we have concrete plans to continue working on extending and applying our MBRE approach and language in the context of other large collaborative projects we will be involved in the near future.

REFERENCES

[1] Richard Van Noorden and Declan Butler. Science in europe: by the numbers. *Nature*, 569, May 2019.

[2] European Commission-DG CONNECT. Digital economy and society index (DESI) 2020 - the EU ICT sector and its R&D performance. Online at https://ec.europa.eu/newsroom/dae/document.cfm?doc_id=72352. Accessed: 2021-04-07.

[3] European Commission. EU research programmes. https://ec.europa.eu/info/funding-tenders/funding-opportunities/funding-programmes/overview-funding-programmes_en. Accessed: 2021-04-07.

[4] IEEE/ISO/IEC. International standard - systems and software engineering – life cycle processes – requirements engineering. Technical Report 29148-2018, IEEE Standards Association, 2018.

[5] Betty H. C. Cheng and Joanne M. Atlee. Research directions in requirements engineering. In *2007 Future of Software Engineering*, FOSE '07, page 285–303, USA, 2007. IEEE Computer Society.

[6] S. Assar. Model driven requirements engineering: Mapping the field and beyond. In *2014 IEEE 4th International Model-Driven Requirements Engineering Workshop (MoDRE)*, pages 1–6, 2014.

[7] Marco Brambilla, Jordi Cabot, and Manuel Wimmer. Model-Driven software engineering in practice. *Synthesis Lectures on Software Engineering*, 1(1):1–182, 2012.

[8] Accelopment. Lessons learnt from horizon 2020 for its final 2 years. https://accelopment.com/blog/lessons-learnt-from-horizon-2020-for-its-final-2-years/, February 2019. Accessed: 2021-04-07.

[9] Ecsel joint undertaking work plan 2020. https://www.ecsel.eu/calls-2020-wp2020.

[10] ECSEL-JU. Productive4.0 project. https://www.ecsel.eu/projects/productive40. Accessed: 2021-04-07.

[11] Seda Gürses, Magali Seguran, and Nicola Zannone. Requirements engineering within a large-scale security-oriented research project: lessons learned. *Requirements Engineering*, 18(1):43–66, Mar 2013.

[12] Daniel Nepelski and Giuseppe Piroli. Organizational diversity and innovation potential of EU-funded research projects. *J. Technol. Transf.*, 43(3):615–639, June 2018.

[13] Andrey Sadovykh, Alessandra Bagnato, Arne J. Berre, and Stale Walderhaug. Archimate as a specification language for big data applications - databio example. In Jean-Michel Bruel, Manuel Mazzara, and Bertrand Meyer, editors, *Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*, pages 191–199, Cham, 2020. Springer International Publishing.

[14] Andrey Sadovykh, Tewfik Ziadi, Alessandra Bagnato, Thorsten Berger, Jan-Philipp Steghöfer, Jacques Robin, Raul Mazo, and Elena Gallego. Revamp2 project: Towards round-trip engineering of software product lines - approach, intermediate results and challenges. In Manuel Mazzara, Jean-Michel Bruel, Bertrand Meyer, and Alexander Petrenko, editors, *Software Technology: Methods and Tools*, pages 406–417, Cham, 2019. Springer International Publishing.

[15] Wasif Afzal, Hugo Bruneliere, Davide Di Ruscio, Andrey Sadovykh, Silvia Mazzini, Eric Cariou, Dragos Truscan, Jordi Cabot, Abel Gómez, Jesús Gorroñogoitia, Luigi Pomante, and Pavel Smrz. The MegaM@Rt2 ECSEL project: MegaModelling at runtime – scalable model-based framework for continuous development and runtime validation of complex systems. *Microprocessors and Microsystems*, 61:86–95, 2018.

[16] Axel van Lamsweerde. Requirements engineering in the year 00, 2000.

[17] Eric SK Yu. Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. In *ISRE 1997*, pages 226–235, Washington, DC, U.S.A., 1997. IEEE Computer Society.

[18] Axel Van Lamsweerde. Goal-Oriented Requirements Engineering: A Guided Tour. In *ISRE 2001*, pages 249–262, Washington, DC, U.S.A., 2001. IEEE Computer Society.

[19] Object Management Group (OMG). Requirements Interchange Format (ReqIF), 2020.

[20] B. Baudry, C. Nebut, and Y. L. Traon. Model-driven engineering for requirements analysis. In *11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007)*, pages 459–459, 2007.

[21] Arda Goknil, Ivan Kurtev, and Klaas van den Berg. A metamodeling approach for reasoning about requirements. In Ina Schieferdecker and Alan Hartman, editors, *Model Driven Architecture – Foundations and Applications*, pages 310–325, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[22] Emmanuel Letier, Jeff Kramer, Jeff Magee, and Sebastian Uchitel. Deriving event-based transition systems from goal-oriented requirements models. *Automated Software Engineering*, 15(2):175–206, 2008.

[23] Stefan Karg, Alexander Raschke, Matthias Tichy, and Grischa Liebel. Model-driven software engineering in the openETCS project. In *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*, 2016.

[24] H Solheim, F Lillehagen, S A Petersen, H Jorgensen, and M Anastasiou. Model-driven visual requirements engineering. In *13th IEEE International Conference on Requirements Engineering (RE'05)*, 2005.

[25] OMG Unified Modeling Language. https://www.uml.org/.

[26] OMG Systems Modeling Language (SysML). https://www.omgsysml.org/.

[27] European Cooperation for Space Standardization. Space engineering software. Technical Report ECSS-E-ST-40C, ECSS Secretariat ESA-ESTEC Requirements & Standards Division, Noordwijk, The Netherlands, March 2009.

[28] Thomas D. Nielsen, S. Hovda, A. Fernández, H. Langseth, A. Madsen, A. Masegosa, and A. Salmerón. Requirement engineering for a small project with pre-specified scope. In *NIK*, 2014.

[29] Andreas Koukias, Gökan May, Volodymyr Vasyutynskyy, Drazen Nadoveza, Jessica C. McCarthy, Marco Taisch, and Dimitris Kiritsis. Approach on analysis of heterogeneous requirements in software engineering. *IFAC Proceedings Volumes*, 46(7):372–377, 2013. 11th IFAC Workshop on Intelligent Manufacturing Systems.

[30] MegaM@Rt2 project. Deliverable d2.2 MegaM@Rt2 design tool set specification. https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e5b8e1feb7&appId=PPGMS. Accessed: 2021-04-07.

[31] Modelio: A Collaborative Business or Software Modeling Platform. https://www.modeliosoft.com/en/.

[32] The ArchiMate Enterprise Architecture Modeling Language. https://www.opengroup.org/archimate-forum/archimate-overview.

[33] OMG. Business Process Model and Notation (BPMN). https://www.omg.org/bpmn/.

[34] Andrey Sadovykh, Hugo Bruneliere, and Dragos Truscan. Dataset - Survey results - Applying Model-based Requirements Engineering in Three Large European Collaborative Projects, June 2021.

[35] Karl Wiegers and Joy Beatty. *Software Requirements*. Pearson Education, August 2013.

[36] Andrey Sadovykh, Gunnar Widforss, Dragos Truscan, Eduard Paul Enoiu, Wissam Mallouli, Rosa Iglesias, Alessandra Bagnto, and Olga Hendel. Veridevops: Automated protection and prevention to meet security requirements in devops. In *Design, Automation and Test in Europe Conference (DATE'20)*, 2020.

[37] A Sadovykh, D Truscan, P Pierini, G Widforss, A Ashraf, H Bruneliere, P Smrz, A Bagnato, W Afzal, and A E Hortelano. On the use of hackathons to enhance collaboration in large collaborative projects : - a preliminary case study of the MegaM@Rt2 EU project -. In *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 498–503, March 2019.