Assessment of the SEMCO Model-Based Repository Approach for Software System Engineering

Brahim Hamid

Assessment of the SEMCO Model-Based Repository Approach for Software System Engineering

Brahim Hamid(✉)

IRIT, University of Toulouse, 118 Route de Narbonne,
31062 Toulouse Cedex 9, France

hamid@irit.fr

Abstract. We have developed a methodological tool support for soft-ware development based on the reuse of dedicated subsystems, that have been pre-engineered to adapt to a specific domain. This paper proposes an empirical evaluation of the proposed approach through its practical application to a use case in the railway domain with strong security and dependability requirements, followed by a description of a survey per-formed among domain experts to better understand their perceptions regarding our approach. The case study enables us to determine that our approach centered around a model-based repository of patterns leads to a reduction in the number of steps in the engineering process or to their simplification. The survey assists in assessing whether domain experts agree on the benefits of adopting the model-based repository approach in a real industrial context.

Keywords: Modeling artifact · Model-based repository · Model repos-itory · Metamodel · Model-driven engineering · Software system engi-neering

1 Introduction

Repositories of modeling artifacts have recently gained increased attention as a means of encouraging reuse in software engineering. In fact, repository-centric development processes are more widely adopted in software system development than are other approaches, such as architecture-centric or pattern-centric development processes. According to Bernstein and Dayal [1], a repository is a shared database of information regarding engineered artifacts. In our work, we go one step further: we conceptualize a model-based repository to support the specifications, definitions and packaging of a set of modeling artifacts. During system development lifecycles, modeling artifacts may be used in various forms such as domain models, design patterns, component models, code modules, test and code generators [3]. However, the question remains of when and how to integrate modeling artifacts into the software-intensive system development process. Closely related to our vision is the approach of [8,12] that autonomously locates

DOI: 10.1007/978-3-319-66854-3_9

and delivers task-relevant and personalized components into the current software development environment.

The envisioned modeling framework consists of two main pillars: solid theory and proven principles. The first pillar offers an integrated conceptual design for the specification and development of a model-based repository and its contents; the second pillar offers a concrete and coherent methodology for the development of software systems based on the repository. We have developed a System and software Engineering with Multi-COncerns (SEMCO) framework [5,6] to assist system and software developers in the domain of resource constrained systems to capture, implement and document distributed system applications. SEMCO aims at supporting model- and pattern-based development approaches by defining and providing several artifacts types representing the different related engineering concerns (Security, Dependability, Safety and Resources) and architectural information. The approach relies on an MDE tool suite called SEMCOMDT to support the methodology and thus, in our context, to support the automated construction of and access to the model-based repository. The foundation of SEMCOMDT is a collection of Domain-Specific Modeling Languages (DSMLs) built on an integrated repository of modeling artifacts working as a group, each one relevant to a particular key concern. The resulting tool chain supports two categories of users: developers who reuse existing artifacts from the repository and developers of artifacts to be stored in the repository.

This work provides evidence of the SEMCO benefits and applicability through the example of a representative industrial case from the FP7 TERESA project[1] by applying the approach to Pattern-Based System Engineering (PBSE). Modeling artifacts derived from the model repository and associated with domain-specific models can assist developers in integrating in-development application building blocks with pre-defined modeling artifact building blocks. Empirical evaluation of the proposed approach, using Key Performance Indicators (KPIs), is presented through its practical application to a use case in the railway domain. KPI is an industry term that refers to a type of performance measurement, to evaluate the success of a particular activity, project or product. The second evaluation step is performed using a survey to better understand the perceptions of practitioners regarding our findings. Based on the background of our research project partners', we started with an existing approach, such as the Technology Acceptance Model (TAM) [2]. For instance, we have identified a set of measures to evaluate our solutions, which produced the definition of a set of hypotheses. Then, we enhanced the TAM using the ISO-9126's quality-in-use dimensions, i.e., effectiveness, productivity, safety and satisfaction. In some cases, we employed the factors developed in Rogers' theory of innovation diffusion [9], which were involved in technology adoption: (1) Trialability; (2) Compatibility; (3) Relative advantage; (4) Observability; and (5) Complexity. We designed a research strategy that is focused on testing these hypotheses by performing and reporting empirical studies, following the guidelines described in [11]

[1] http://www.teresa-project.org/.

(e.g., construct a set of research questions, design of questionnaires, data collection, using statistical tools for data analysis and threat to validity).

The remainder of this paper is organized as follows. Section 2 provides theoretical background and SEMCO. In Sect. 3, we present an empirical evaluation of our approach via a railway case study in framework of the TERESA project and a survey. Finally, Sect. 4 presents our conclusions and suggests possible directions for future work.

2 Motivations and Theoretical Background

A software system architect must work at multiple different levels. Integrating all the subsystems and accounting for the associated software requirements in a seamless fashion is quite a challenge given the various critical requirements and uncertainties associated with them. We propose a solution for software development based on the reuse of dedicated subsystems, so-called modeling artifacts that have been pre-engineered to adapt to a specific domain. We use MDE to develop a repository of models and a methodology for developing software systems based on this repository.

MDE promotes models as first-class elements. A model can be represented at different levels of abstraction, and the concept of MDE is based on (1) the metamodeling techniques used to describe these models and (2) the mechanisms used to specify the relations between them. Domain-Specific Modeling (DSM) [4] in software engineering is a methodology in which models are used to specify applications within a particular domain. Several DSM environments exist, including the open-source Eclipse Modeling Framework (EMF) [10] and its extended version Eclipse Modeling Framework Technology (EMFT)[2]. Note, however, that our vision is not limited to the EMF platform. EMF offers a set of tools to specify metamodels in Ecore and to generate other representations of these metamodels.

The proposed approach consists of six main steps (the numbers in parentheses below correspond to those indicated in Fig. 1):

- The first step (1) is the creation of a conceptual model of the repository.
- The second step (2) is the creation of a set of DSMLs for the specification of modeling artifacts.
- Using the DSMLs and the conceptual model of the repository, a software engineering expert defines the model and builds the repository (3).
- Using the DSMLs, a modeling artifact expert, with the help of the system and software engineering expert, defines the modeling artifacts and begins populating the repository (4).
- Then, a domain process expert adapts the modeling artifacts into a form that is suitable for the system development process (5). For instance, they might be adapted for use within a certain development environment.
- Finally, a domain engineer reuses the resulting modeling artifacts that have been adapted and transformed for the given engineering environment (development platform) to develop a domain application (6).

[2] https://eclipse.org/modeling/emft/.

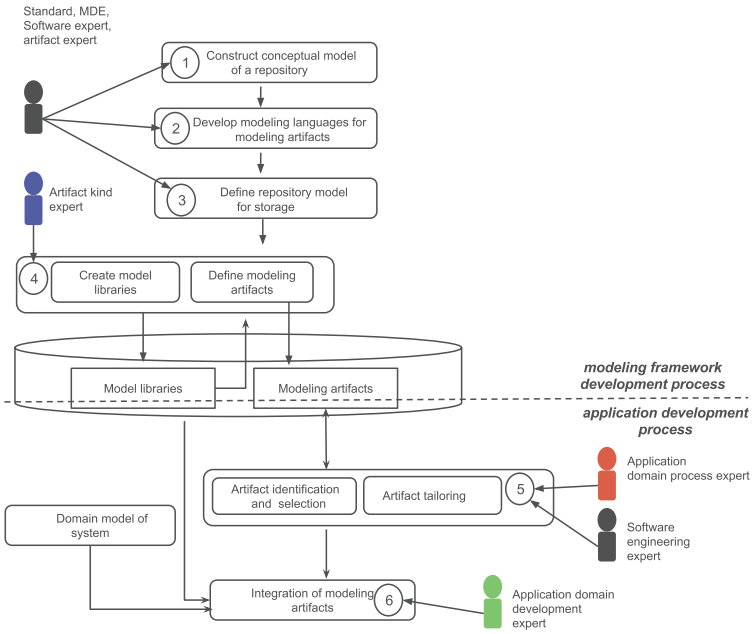

Fig. 1. Methodology for the creation of the model-based repository modeling framework

The first two steps (1 and 2) are performed once for a given set of domains. The inputs to these steps are expertise, standards and best practices from software system engineering. Step 3 is performed once for a given set of domains. Step 4 is performed once per application domain. Step 5 is performed once for each development environment. Performing Step 4 requires knowledge of software engineering, whereas Step 5 requires knowledge of both software engineering and the system development process for a specific application domain. Step 6 is performed once for every system in the application domain. This step requires the availability of knowledge of the specific target system and dedicated tools that are customized for a given development platform.

3 Empirical Assessment

In this section, we first report on an industrial case study performed in the railway domain (Sect. 3.1) and then present a description of a survey performed among TERESA domain experts (Sect. 3.2).

3.1 Case Study

Our case study is aimed at investigating the feasibility of our approach and the level of effort involved in its application using Key Performance Indicators (KPIs). A performance indicator or key performance indicator (KPI) is an industry term referring to a type of performance measurement. KPIs are commonly used to evaluate the success of a particular activity, project or company. In the context of the TERESA project, we evaluated our approach in the construction of an engineering discipline that is adapted to RCES by combining MDE and a model-based repository of S&D patterns and their related property models.

3.1.1 Case Selection

The *Safe4Rail* system is composed of (1) a *Clock* which generates a periodic event to trigger the system to estimate the current position and speed and to supervise that the train complies with the current track restrictions, (2) a *Environmental Conditions* to represent the physical interaction between environment (train, track, others) with the sensors of the system, (3) a *Balise* which represents a balise installed on the track to supply to the train supervision system with new information regarding the current position and the track conditions, (4) a *Safe Train Interface* which represents the actuators for the application and (5) a *Supervision System*. In turn, the *Supervision System* is composed of (a) a *BaliseReader* to detect and read the information provided by the balise on the rail, (b) a *Supervision*, as the main component of the system, which is responsible of carrying out the functionality of the system and (c) *Sensors* to provide the actual position and speed of the train and the track conditions to the system.

To illustrate our study, two different railway industry scenarios will be considered: (1) Railway manufacturing group which is divided in subsidiary companies specialized in the development of train and railway infrastructure systems and (2) An SME that develops safety related embedded systems. This company has a proven experience in the development of safety related embedded systems, but accounts scarce experience in the railway domain. Therefore, the estimations given for the railway domain provide two values with associated argumentation:

1. *Large and complex safety systems*, such as on-board ERTMS/ETCS, and
2. *Intermediate safety systems*, such as traction control safety supervision.

3.1.2 Research Questions

The purpose of our study was to address the following two research questions:

- **RQ1:** *Does the proposed approach reduce the effort involved in developing a new application (design and implementation)?* To answer **RQ1**, we evaluate whether the approach leads to a reduced number of steps in the engineering process or to their simplification, and we assess whether the domain experts agree on the benefits of adopting the approach in a real industrial context for the development of new applications.
- **RQ2:** *Is the effort involved to engineer a new version of an existing application to add a security or dependability property acceptable?* For **RQ2**, we measure the ability of the approach to integrate security and dependability solutions into existing products.

3.1.3 Data Collection Procedure

The procedure used for developing the case study closely followed the approach described in Sect. 2. Given a repository requirements, a conceptual model is built that fulfill these design requirements. The next step is the creation of a set of Domain-Specific Modeling Languages (DSMLs) for the specification of modeling artifacts. In the context of our experiment, we have developed System and

Software Engineering Pattern Metamodel (SEPM) [7] and Generic PRoperty Metamodel (GPRM) [13] to model pattern and property models, respectively. Using the DSMLs and the conceptual model of the repository, a software engineering expert defines the model and builds the repository. This work was done by the author. Then, the modeling artifact expert, with the help of the system and software engineering expert, defines the modeling artifacts and begins populating the repository. Then, a domain process expert adapts the modeling artifacts into a form that is suitable for the system development process. Finally, a domain engineer reuses the resulting modeling artifacts that have been adapted and transformed for the given engineering environment to develop a domain application.

For the purpose of our study, we have developed SEMCOMDT [3](SEMCO Model Development Tools) as an MDE tool chain to support all steps of our approach using EMFT. All metamodels used are specified using EMF. To create model instances of the proposed metamodel, we choose to use a tree-structured concrete syntax provided by EMFT. It provides graphical, but not-diagrammatic notations, to specify Ecore models. The SEPM's concrete syntax is described using a mixed syntax combining structured-tree syntax and a UML-based diagrammatic syntax. The structure of the repository is derived from the repository structure model and implemented using Java and the Eclipse CDO[4] framework. This work was done by the two Phd students.

In the following, KPIs are presented with an estimated target value submitted at the beginning of the case studies, and real values are estimated upon completion of the case studies. Both values are averages because the individual values highly depend on certain factors, such as the size of the project, the product requirements, the expertise of the engineering team, the use of the approach facilities and the availability of the appropriate patterns. In both cases, KPIs are estimated based on previous knowledge of implementing such systems or equivalents. A description of the considered constraints and assumptions is discussed below.

- *(K1) Overall engineering cost.* Cost to develop a new application (design and implementation) with dependability requirements using a target reduction of 10% to 20% (on average).
- *(K2) Percentage of reused code.* Amount of code reused from existing patterns in a new development with dependability requirements using a target reduction of 20% to 60% (on average).
- *(K3) Engineering time.* Time to develop a new application (design and implementation) with dependability requirements using a target reduction of 10% to 25% (on average).
- *(K4) Re-engineering time.* Time to engineer a new version of an existing application to add a dependability property with an estimated target reduction of 40% (on average).

[3] http://www.semcomdt.org.
[4] http://www.eclipse.org/cdo/.

– *(K5) Errors in reused code.* Number of errors appearing in code reused from patterns using a target value of 10 as a factor of the probability of errors in reused code with respect to new code.
– *(K6) Code quality.* Readability and compliance of the reused code with the required standard with an estimated target value of compliance of 100%.
– *(K7) Maintenance cost.* Total cost and effort associated with bugs being detected after deployment using a target reduction of 10% to 20% (on average).
– *(K8) Incident response.* Time and effort for identifying affected products from reused code/concepts with an estimated target reduction of 30% to 50% (on average).

3.1.4 Case Study Results and Discussion

Here, we discuss the results of the case study focusing on answering the research questions that we presented in Sect. 3.1. The creation of the conceptual model of the repository required approximately 4 person months. The creation of the DSMLs for the modeling artifacts took approximately 6 person months. The construction of the model of the repository and the implementation of the MDE tool chain took 6 person months. The construction of the domain model took another 3 person months. The process of populating the repository took one month. The proposed tool chain is designed to support the proposed metamodels, and hence, the tool chain and the remainder of the activities involved in the approach may be developed in parallel. This activity needs to be performed only once for a given set of domains. We expect the effort for the creation of a DSMLs and the development of tools to be less on future applications, as we had to address several technical details in relation to using EMFT and CDO in our first application.

A comparison of the resulting KPIs in the two cases is shown in Fig. 2, where the estimated KPI values for the newly proposed approach at the end of the case study are presented. From this comparison, it appears that using the security and dependability pattern-based approach brings significant advantages in S&D engineering, especially for intermediate safety systems.

RQ1. *Does the proposed approach reduce the effort involved in developing a new application (design and implementation)?* With regard to the overall engineering cost, we estimate that the development of large and complex systems (ERTMS/ETCS), respectively intermediate systems, is reduced by an average of 12.5%, respectively 30% (**K1**). The overall engineering cost is reduced as follows. During the safety concept, system architecture, software architecture and module detailed design phases, a reduction in the time to formalize and document the design is observed using already-developed design patterns that include all necessary safety information and reducing the effort required to document detailed descriptions by hand. Moreover, a reduction in the time and effort associated with verification is also observed because the design patterns are already verified and provide a common understanding for both designers and verifiers. Finally,

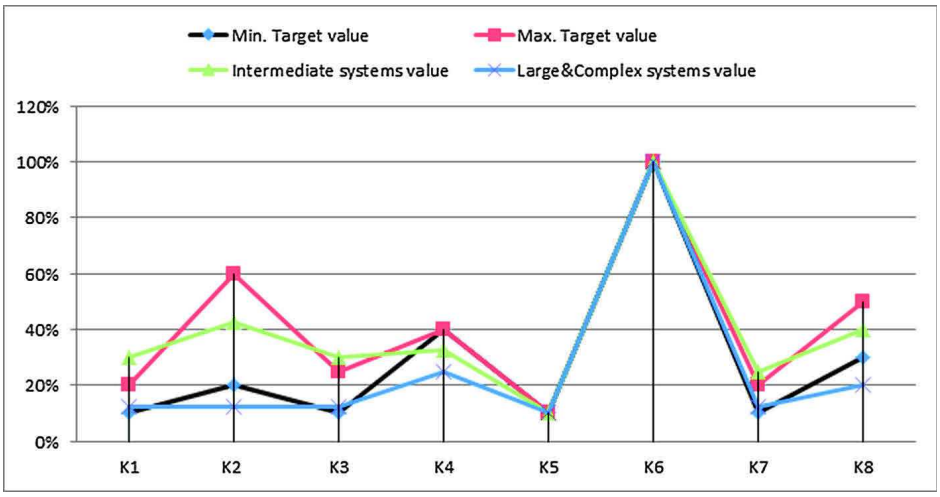

Fig. 2. Intermediate, large & complex safety systems

a reduction in the time and effort associated with RAMS[5] analysis is observed because the design patterns are already verified and provide a common understanding for both designers and RAMS engineers. With regard to the percentage of reused code, we estimate that during the development of large and complex systems (ERTMS/ETCS), resp. intermediate systems, code is reused at an average of 12.5%, respectively 43% (**K2**). The selected case study, namely, ERTMS/ETCS on-board railway signaling acting as a large and complex safety system, is a representative SIL4 safety embedded system in which multiple design patterns can be instantiated. However, ERTMS/ETCS is a highly complex system in which most of the software application implements the safety and functional requirements established by the standard for interoperability. Selected and integrated design patterns provide the safety skeleton and safety architectural foundation of the system (key foundation), where the system-specific application is deployed and executed. However, the ratio between system-specific software and software that can be provided by design patterns is less than 0.1.

This is a paradox because other safety subsystems of the train that perform intermediate complexity safety functions might be developed with a much smaller set of design patterns, although the ratio of design-pattern-based safety software compared to system-specific safety software might be at least one to one. For example, the safety function of a railway traction system (SIL2) acting as an intermediate safety system has an intermediate complexity. It must compare already acquired current, voltage and temperature measurements with given minimum and maximum thresholds and perform a small set of coherency checks on the measurements.

RQ2. *Is the effort involved to engineer a new version of an existing application to add a security or dependability property acceptable?* With regard to the re-engineering, we estimate that the development of large and complex systems (ERTMS/ETCS), respectively intermediate systems, is reduced by an average

[5] Stands for Reliability, Availability, Maintainability, and Safety.

of 25%, respectively 33% (**K4**). After developing the demonstrator and considering the previous estimation of **K2** (percentage of reused code) and that safety design patterns provide key foundational patterns for the development of safety systems, we estimate that the maintenance cost is reduced in large and complex (ERTMS/ETCS), resp. intermediate systems, by an average of 12.5%, respectively 25% (**K7**). Moreover, we estimate that the time and effort for incident response is reduced in large and complex (ERTMS/ETCS) systems, resp. intermediate systems, by an average of 20%, resp. 40% (**K8**). An example is the reduction of complex incident responses associated with the operation of safety replicas (e.g., data agreement, safety communication layer, etc.) that require a considerable amount of time to be analyzed and solved.

3.2 Survey

After the completion of our case study, we conducted an experiment where we presented our approach and the solution of our case study in order to collect feedback from industry practitioners through a survey. The case study enables us to determine that the model-based repository approach leads to a reduced number or to a simplification of the engineering process steps, whereas the survey assists in assessing whether domain experts agree on the benefit of adopting the proposed approach in a real industrial context. In the following, we present and discuss the design and results of this survey.

3.2.1 Context and Description of the Methodology for Experimentation

The approach, its corresponding tool suite and the solution of our case study presented in Sect. 3.1 were proposed to the industry practitioners for evaluation through a survey. The purpose of this survey is to give an overview of the design of a software architecture of a small but sufficiently complex system that we will use to illustrate the model-based repository approach we propose.

All participants attended the TERESA MDE workshop in Toulouse, where the experiment was initiated. Six security experts (SEC) participated in the survey: four from the TERESA domains and two from other domains. Five dependability experts (DEP) also participated in the survey: three from the TERESA domains and tow from another domain. In addition, five software engineering (SEN) experts participated in the survey: two from the TERESA domains and three from other domains. All participants were recognized as experts in their domains with a high level of skill in security or dependability who had already participated in the development of several projects related to their skills. All participants were already familiar with S&D patterns and modeling tools. Overall, 62% of the participants had over two years of S&D engineering experience and a further 19% had at least one year of experience with S&D engineering.

The objective of the survey was to determine whether domain experts agree on the benefits of adopting our approach in a real industrial context. For that,

we proposed to use the factors developed in Rogers' theory of innovation diffusion [9], involved in technology adoption: (1) Trialability; (2) Compatibility; (3) Relative advantage; (4) Observability; and (5) Complexity. To address these factors we have developed a simple questionnaire, containing 20 questions (some questions were used to measure more than one factor), to capture the information required. Precisely, we focus on (1) the perceived usefulness of the approach itself; (2) the perceived usefulness of the conceptual models, the execution of the approach and the tool suite as a means of building and reusing modeling artifacts; and (3) the willingness to use a model-based repository approach in future related activities. Next, the participants were asked to scale their satisfaction on a scale from 1 to 5, 1 being the lowest value of satisfaction or the greatest difficulty (meaning *Not Useful at All*, *Very Difficult*, *Very Probably Not*) and 5 being the highest value of satisfaction regarding the presented concepts or the greatest ease in realizing a solution to the given question (meaning *Extremely Useful*, *Very Easy*, *Definitely*). We calculated the averages of the values provided by the participants.

3.2.2 The Questionnaire

The questionnaire was divided into three parts. The first section, which consisted of *Q1–Q5*, as shown below, concerned the perceived usefulness of the approach itself.

- *Q1: Do you think that a model-based repository approach is a good idea?*
- *Q2: Do you think that a model-based repository approach helps to maintain focus without being distracted by other aspects of software engineering?*
- *Q3: Is the information provided by the model-based repository approach useful for defining meaningful 'units of solution'?*
- *Q4: Does the model-based repository approach reduce the effort involved in developing a new application (design and implementation)?*
- *Q5: Do you think that a model-based repository approach avoids the reinvention of existing solutions?*

The second section, *Q6–Q14*, addressed the conceptual models, the execution of the approach and the tool suite as a means of building and reusing modeling artifacts.

- *Q6: Were the presented conceptual models easy to understand?*
- *Q7: Overall, how easy to follow were the steps of our approach?*
- *Q8: Was the presented tool -suite easy to use?*
- *Q9: Did you find the models easy to use for engineering your application?*
- *Q10: Did you find it easy to define new security and dependability patterns?*
- *Q11: Do you think that the tool provided for tailoring security and dependability patterns is easy to use?*
- *Q12: Do you think that the tools provided for the integration of security and dependability patterns is easy to use?*

– *Q13: How easy was it to integrate the tool suite into your favorite development environment?*
– *Q14: Does the presented tool suite provide useful assistance in the development of secure and dependable applications?*

The last section, *Q15–Q20*, concerned the participants' willingness to use a model-based repository approach in future related activities.

– *Q15: Would you see value in adopting the presented approach at your company?*
– *Q16: Would you like to define other kinds of reusable modeling artifacts in the future?*
– *Q17: Would you like to install other SEMCO plugins in the future?*
– *Q18: Would you like to use the approach in the future?*
– *Q19: Would you like to customize various SEMCO plugins in the future?*
– *Q20: Would you like to extend various features of the approach in the future?*

3.2.3 Experiment Conduct

Because of the technical and administrative problems encountered in the design, implementation and deployment of repository software, we performed this group of tasks in a pre-processing step prior to the study. Thus, in our experiment, we considered only a pre-defined model-based repository. In our case, the repository server application is hosted on a machine at the University of Toulouse. Once the service was initialized, we proceeded to the repository initialization. A Java-based GUI application enabled the creation of the repository structure as well as the initialization and management of compartments and users. Each of the participants was given a login and password to manage her/his artifacts. The second element of the SEMCO environment is the client tool suite, which is provided as a set of Eclipse plugins. The client tool suite consists of a set of modeling artifact editors, helpers for depositing artifacts into and retrieving them from the repository, and the required repository interfaces. The installation is finalized by setting the preferences on the client to point to the correct repository. The experiment included five tasks: SEMCO plugin installation, property model development, pattern development, pattern tailoring and pattern integration.

For each task, a set of materials was provided to the participants at the beginning of the experiment: the SEMCO tool suite and its accompanying installation and user documentation, a detailed textual description of the patterns and their properties, a detailed requirements document, and a conceptual model of the system under development in the form of UML diagrams. Finally, the participants were given a subjective post-experiment questionnaire consisting of a set of questions, as described previously, and space for comments. In addition, they were provided with a sheet presenting the instructions for each task (e.g., what properties to specify and what patterns to develop, when to take note of the time, and so on).

Training of the Participants. During the TERESA MDE workshop in Toulouse, the participants were trained to use the method and the SEMCO tool suite.

The training was conducted in two sessions on the same day. The first session was 3 h long and was managed by two instructors. The first instructor introduced MDE and presented a pattern-based development methodology and how it might be used to support the development of secure and dependable applications. In addition, a 1-h practice session on Eclipse and the EMFT environment was presented by the second instructor as a laboratory exercise. During the second hour-long session, several operating examples were introduced to the participants, with detailed explanations, by two additional instructors who participated in the development of the SEMCO tool suite.

Execution of the Experiment. The experiment was conducted in the context of the TERESA project regarding the development of secure and dependable applications. The participants were grouped according to whether they possessed expertise in the related domains. To improve and simplify the evaluation of the results, the participants were asked to use only tools and methods that were presented during the training sessions. However, they were allowed to use their own resources, predominantly those related to the description of patterns in their domain. Before they started, a general description of the objective of the study was presented (30 min). Portions of these evaluation studies were performed internally, whereas other studies were outsourced. The participants were given 2 h on site to complete the installation tasks (performed during the TERESA MDE workshop in Toulouse). Then, a 6-month outsourced evaluation was conducted to complete the other tasks.

Data Collection. The questionnaire was uploaded online using Google Docs, and the link to the questionnaire was forwarded to the participants. All of the registered participants received the questionnaire link in this manner. At the time of the survey, twelve participants were members of the TERESA project. The data collection process began in January 2013 and continued for 6 months, and ultimately, a sample of sixteen usable responses was collected. As the first step of the analysis, the mean scores corresponding to the participant's responses were calculated.

3.2.4 Survey Results and Discussion

After the data were collected, data analysis was conducted to determine the answers to the research questions. The following presents an overview of the results of our experiment. The purpose of the first question was to assess the first impressions of the participants. This answer was used as a baseline and as a measure of whether they were prepared and motivated to perform the subsequent steps of the experiment.

Of the participants in the experiment, 75% of them thought that the approach would be extremely useful, and the remaining 25% thought that the approach would be very useful. After being presented with a high-level description of the approach, 31% of the participants perceived the approach as being extremely useful for maintaining focus without being distracted by other aspects of software engineering, whereas 69% thought it would be very useful. Based on the col-

lected responses, 56% of participants found the approach very useful for defining meaningful units of solution, and the remaining 44% found it extremely useful. We can also conclude that 69% of the participants found the approach to be very useful for development through reuse, and 75% of them found the approach to be very useful for development for reuse.

When presented with the conceptual models of the approach, 50% of the participants perceived the approach as being very easy to understand, whereas the remaining 50% thought it was easy to understand. The results reveal also that 31% of the participants perceived the approach as very easy to follow, whereas 56% thought it was easy, and the remaining 13% experienced average difficulty. Regrading the tool support, 51% of the participants found the tool suite very easy to use; meanwhile, 19% experienced average difficulty in using the tool suite. With regards to integrating the provided tool-suite into other development environment, 38% of the participants indicated that it is seldom easy, 56% experienced average difficulty in integrating the tool-suite, whereas the remaining participants (6%) found it difficult. However, the current tool suite based on EMFT was expected to be useful. A total of 31% of the participants believed that the tool-suite would be extremely useful for the development of secure and dependable applications, and a further 44% thought that it would be very useful. The remaining 25% of the participants thought that the tool suite would be useful.

With regards to the adoption of the approach, 63% of the participants thought that there was definitely value in adopting the approach, and a further 31% thought that the approach was very probably worth adopting. Regarding the extent to which the participants indicated that they would likely use the model-based repository approach in the future for the development of other kinds of systems, 19% Of the them indicated that they would definitely continue their application of the approach, whereas 63% said that they would very probably do so.

In summary, the answers received in our survey suggest that the proposed approach was overall regarded as easy to learn and to follow. Moreover, the participants thought that it would be beneficial to use within their context. These responses indicate that model-based repository approaches to the development of software systems should be investigated further. We believe that these experimental results may be generalized to other pattern-based development approaches and, in a broader scope, to other model-based development approaches, as the patterns were provided as models for the application developers.

4 Conclusion and Future Work

The proposed model-based approach for software application development relies on a repository of models and focuses on the problem of software system engineering through a design philosophy that fosters reuse. This approach was evaluated in the context of the TERESA project for application to a repository of S&D

patterns and property models. Following the specification, design, implementation and deployment of an S&D pattern repository, pattern designers can define property and pattern models and store them in the repository. System designers can then reuse existing patterns from the repository through identification and tailoring mechanisms, leading to simpler and more seamless designs with higher quality and reduced cost. By means of the practical demonstration provided by our case study, we can validate the feasibility and effectiveness of the proposed specification and design frameworks. We also conducted a survey of industry practitioners among TERESA members and other security, dependability and software engineering experts. The preliminary evidence indicates that users are satisfied with the notion of a development approach centered around a model-based repository of patterns and, in a broader context, a model-based repository of modeling artifacts. However, the results also highlight one of the main challenges, namely, the design of an automated search functionality to allow the user to derive the necessary modeling artifacts from an analysis of the requirements for a project.

In our future work, we plan to study the automation of the model search and tailoring tasks. Our vision is for modeling artifacts to be inferred from the browsing history of users and constructed from a set of already developed applications. We would also like to study the integration of our tools with other MDE tools. For that purpose, we need to implement other kinds of software and means of generating validated artifacts, such as programming language code and certification artifacts, that are capable of producing a restrictive set of artifacts that comply with domain standards.

References

1. Bernstein, P.A., Dayal, U.: An overview of repository technology. In: Proceedings of the 20th International Conference on Very Large Data Bases, VLDB 1994, pp. 705–713. Morgan Kaufmann Publishers Inc. (1994)
2. Davis, F.: Perceived usefulness, perceived ease of use, and user acceptance of information technology. MIS Q. **13**(3), 319 (1989)
3. Frakes, W., Kang, K.: Software reuse research: status and future. IEEE Trans. Softw. Eng. **31**(7), 529–536 (2005)
4. Gray, J., Tolvanen, J.-P., Kelly, S., Gokhale, A., Neema, S., Sprinkle, J.: Domain-specific modeling. In: Fishwick, P. (ed.) Handbook of Dynamic System Modeling, Chap. 7, pp. 1–20. Chapman & Hall/CRC, Boca Raton (2007)
5. Hamid, B.: A model-driven methodology approach for developing a repository of models. In: Ait Ameur, Y., Bellatreche, L., Papadopoulos, G.A. (eds.) MEDI 2014. LNCS, vol. 8748, pp. 29–44. Springer, Cham (2014). doi:10.1007/978-3-319-11587-0_5
6. Hamid, B.: Modeling of secure and dependable applications based on a repository of patterns: the SEMCO approach. Reliab. Digest, IEEE Reliab. Soc. Special Issue Trustworthy Comput. Cybersecur. **1**(1), 9–17 (2014)

7. Hamid, B., Gürgens, S., Jouvray, C., Desnos, N.: Enforcing S&D pattern design in RCES with modeling and formal approaches. In: Whittle, J., Clark, T., Kühne, T. (eds.) MODELS 2011. LNCS, vol. 6981, pp. 319–333. Springer, Heidelberg (2011). doi:10.1007/978-3-642-24485-8_23. ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS)
8. Katalagarianos, P., Vassiliou, Y.: On the reuse of software: a case-based approach employing a repository. Autom. Softw. Eng. **2**(1), 55–86 (1995)
9. Rogers, E.: Diffusion of Innovations, 5th edn. Free Press, New York (2003)
10. Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.: EMF: Eclipse Modeling Framework 2.0, 2nd edn. Addison-Wesley Professional (2009). ISBN 0321331885
11. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B., Wesslén, A.: Experimentation in Software Engineering: An Introduction. Kluwer Academic Publishers, Norwell (2000)
12. Ye, Y., Fischer, G.: Reuse-conducive development environments. Autom. Softw. Eng. **12**(2), 199–235 (2005)
13. Ziani, A., Hamid, B., Trujillo, S.: Towards a unified meta-model for resources-constrained embedded systems. In: 37th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), pp. 485–492. IEEE (2011)