Enhance Your Model-driven Modernization Process with Agile Practices

Sylvia Ilieva[1], Iva Krasteva[2], Gorka Benguria[3] and Brian Elvesæter[4]

[1] IICT-BAS, Acad. G. Bonchev str., bl.25A, Sofia 1113, Bulgaria

[2] Rila Solutions EAD, Acad. G. Bonchev str., bl.27, Sofia 1113, Bulgaria

[3] TECNALIA R&I, Ed 202, Zamudio 48170, Spain

[4] SINTEF ICT, P. O. Box 124 Blindern, N-0314 Oslo, Norway

Abstract. Cloud computing is currently recognized as an important platform technology that enables service clouds supporting the Software as a Service (SaaS) paradigm. This paper presents the REMICS Methodology, which combines the model-driven approach with agile practices, for the migration of legacy applications to service clouds. The paper gives an overview of the methodology and its application in one case study, the issues faced by three case study providers in applying the methodology and the results of a questionnaire of the applicability of agile practices in addressing the issues.

1 Introduction

Cloud computing and service-oriented architecture (SOA) are recognized game-changing technologies for a cost-efficient and reliable service delivery supporting the Software as a Service (SaaS) paradigm that enables flexible license payment schemas and moves the infrastructure management costs from consumers to service providers. However, building a cloud service system from scratch may require a huge investment in time and efforts. Moreover, legacy systems are difficult to reuse due to platform, documentation and architecture obsolescence.

Model-driven approaches such as the Object Management Group Model Driven Architecture (OMG MDA) and related efforts on domain-specific languages that emphasize the use of formal models in the software development process are gaining popularity. Furthermore, agile methods and techniques are widely adapted and successfully applied in many software development efforts today.

This paper presents an agile model-driven modernization methodology, the REMICS Methodology, for the migration of legacy applications to cloud and SOA. The work is motivated by the need to migrate stand-alone legacy applications that still is of key business value to service cloud systems. Such migration implies advantages in terms of easier maintenance, better accessibility and wider distribution to a range of different users.

Ilieva S., Krasteva I., Benguria G. and Elvesæter B..
Enhance Your Model-driven Modernization Process with Agile Practices.
DOI: 10.5220/0004588800950102
In *Proceedings of the 1st International Workshop in Software Evolution and Modernization* (SEM-2013), pages 95-102
ISBN: 978-989-8565-66-2

The remainder of the paper is structured as follows. Section 2 provides a background summary on related work. Section 3 outlines the initial REMICS Methodology and presents the issues faced by three case study providers in applying it. Section 4 describes the revised agile REMICS Methodology and the results of a questionnaire of applicability of the agile practices. Section 5 presents how the agile methodology is applied in one of the case studies. Finally, Section 6 concludes the paper.

2 Related Work

Comella-Dorda et al. [1] give a survey of legacy system modernization approaches. They distinguish between two main types of modernization: white-box and black-box modernization. White box modernization requires an understanding of the internal parts of a system, and involves re-architecting and re-implementing the system. Black-box modernization is only concerned with the input/output, i.e. the interfaces, of the legacy system, and is often based on wrapping. The REMICS Methodology can be seen as a model-driven black-box modernization approach with flavours of white-box migration, in particular in understanding and transforming the legacy data formats.

Within the OMG, the Architecture-Driven Modernization (ADM) task force [2] is working on standards to support legacy modernization, such as metamodels for knowledge discovery, software visualisation and refactoring.

There are also a number of methods developed in various projects. One of them is the XIRUP process for modernization developed in the MOMOCS project [3]. The method relies on models and transformations but does not include services and SOA.

Nowadays, agile methods and techniques are being widely adapted and successfully applied in many business and application domains. The combination of Model-Driven Development (MDD) and agile software development (known as Agile MDD (AMDD)), is broadly researched and applied ([4], [5] [6]). Although there are many existing SOA methodologies, only few were found to be specifically concerned with incorporating agile software development (e.g. [7]). The mScrum4SOSR is one such methodology, proposed by Chung et al [8]. The methodology extends Scrum with UML modelling and XP techniques in order to provide a comprehensive service-oriented software reengineering process model. Another similar approach is the iterative development approach Model-driven Rapid Development Architecture (SMRDA), which unifies SOA and MDD in order to enhance the efficiency of the development efforts and the reusability of the developed services [9].

Based on the literature review we conducted, there are few methodologies that combine agile software development with both MDD and SOA for software reengineering. Moreover, we are lacking agile methodologies that specifically address model-driven modernization to service clouds.

3 Problems of Modernization Projects

Before elaborating the problems of modernization projects faced in the case studies,

we shortly present the initial methodology that was applied in order to provide information about the context of the experience.

The initial REMICS Methodology focused on the migration of legacy applications to the new cloud infrastructures. The initial release was heavily based on the application of tools along the main activity areas of the methodology. These activity areas define the phases of the migration process.

- **Requirements and Feasibility:** Evaluating feasibility of migration and taking decisions on whether to migrate and what to include in the process.
- **Recover:** Understanding the legacy in terms of functionality, data, constraints, quality of service, and the structure of components or the software architecture.
- **Migrate:** Developing the new architecture and modernizing components, implementing new ones or identifying external services and composing services according to the new architecture.
- **Validate:** Testing and checking the quality of the new system, the coverage of the legacy features and the changes introduced in the migration
- **Control and Supervise:** Provide elements to control the performance of the system and to modify that performance.
- **Withdrawal:** Finalizing it or moving to another cloud infrastructure.

An additional cross-cutting activity area **interoperability** introduced tasks along all the phases with tool support to solve interoperability problems with 3rd party providers or any external components and services.

The initial methodology provided limited support in the sequencing of these activity areas and suggested organizing them in phases following the OpenUp process[1]. Fig. 1 shows a simplified illustration of the phases and their overall sequence as it was provided in the initial methodology.

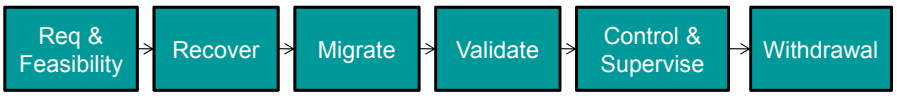

Fig. 1. The REMICS Methodology phases (modified from [12]).

Next we present a summary of the problems identified in the first phases for the different use cases. The three case studies (CS) in the project differ in business domain and technologies used in the legacy systems. The CS1 provides a part of a financial legacy system for generation of financial reports from accounting history, written in COBOL. The source code was written decades ago and has been extended step by step without having refactoring in mind. The CS2 provides a use case "Excursion Management" from a reservation system. The system is written in PL/SQL and Delphi and does not provide web interface. The CS3 provides part of a software product for management of an auto park. The part provided includes administration and maintenance of vehicles in the auto park. The first version of the product was delivered about 10 years ago when it was not intended to serve diverse clients. The code is written in Java and has not undergone any significant refactoring since the

[1] http:// epf.eclipse.org/wikis/openup/

beginning. All of the three case studies started moving their systems to SaaS paradigm according to the suggested traditional sequential approach. Following is a brief description of the problems they faced during the first phases of the modernization process.

During the ***requirements*** identification, case studies reported similar problems. The most challenging issues were: a) identification of new requirements due to lack of system and/or business domain knowledge; b) specification of components for the new cloud deployment models; c) change and evaluation of requirements specification during the project execution and d) definition of validation criteria for the new requirements.

The ***recovery*** phase posed different problems depending on the technologies present in the legacy systems. The CS1 reported that the GOTO statements of the COBOL code worsened the recovery substantially. The granularity of the recovered models was too fine and the high source code volume results in large recovery models. High degree of manual intervention was required including expert knowledge in COBOL, UML and the business domain. The CS2 reported that the automated recovery of the legacy code resulted in loss of data and functionality, which had to be recovered manually. Another problem with medium impact was that the team did not understand the generated UML models due to lack of knowledge. The CS3 also reported that the lack of knowledge of the used recovery tools and standards were one of the problems with highest impact on the project. The case study reported that they had to do refactoring of the legacy code so that the tool can properly recover it.

Only the CS2 and CS3 have passed the migration phases so far. They both reported similar problems during the ***migration***. One main issue was the need of interdisciplinary knowledge of tools and standards in the team. Moreover, both case studies reported the need of handling new emerging requirements for their user interfaces, which were not identified in sufficient details in advance, to keep them as similar to the legacy one as possible.

The new agile methodology is supposed to address the problems faced by the case study providers while applying the initial REMICS Methodology. It is intended to improve the way modernisation is performed by responding to the needs of the industry projects provided by the REMICS case studies.

4 Applicability of Agile Practices

The agile REMICS Methodology has been designed following both top-down and bottom-up approaches. First, five steps were executed to specify the initial modernization methodology in a top-down manner. The aim of the initial methodology is to provide specification of a set of agile practices and techniques, which can be used by modernization projects. The proposed agile practices and techniques have been evaluated to address general challenges of modernization projects. The five steps that were followed for the definition of the initial agile REMICS Methodology are: Identify, Analyze, Select, Define and Formalize. For detailed presentation of the agile REMICS Methodology please refer to [10], [11]. The agile REMICS Methodology proposes a particular implementation of Scrum [13] for large projects. The methodol-

ogy has been adjusted for the characteristics of the REMICS project by defining a number of Modernization Scrum types. Each Scrum type defines Sprint types with particular activities based on the initial REMICS activity areas presented in Section 3. The five Modernization Scrum types are Requirements (RqS), Recovery (RcS), Migration (MgS), Validation (VdS) and Supervise (SpS) (Fig. 2). The interoperability activities are added to the related Sprint types. In addition to the Scrum types, a number of agile development techniques have been adjusted for the extensively used model-driven engineering activities in the REMICS project.

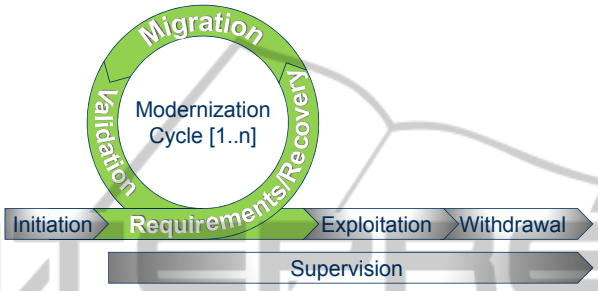

Fig 2. Modernization Scrum types.

Following a bottom-up approach the initial agile REMICS methodology was evaluated and adapted according to the feedback provided by case study providers. A questionnaire on whether particular agile practices are applicable in each of the case study settings was conducted. The three case study providers are either moderately or very familiar with both Scrum and XP methods so their judgement could be considered adequate. As well, the questionnaire surveyed the motivation for applying given practice by answering which particular problem(s) it could address. The overall results showed that most of the suggested agile practices of the initial methodology were applicable in the three case study projects. The results are shown in Table 1. Practices *Modelling by two*, *Continuous Modelling* and *Collective Model Ownership* are customized by Pair programming, Continuous integration and Collective code ownership practices of XP as described in [10]. The *Modernization team* practice is based on the Scrum team practice with a particular specialization of team members due to the diverse and special skills needed for activities in the Scrum types. However, the team acts as a whole and there is no definition of which role can perform certain activity. The team can be seen as a set of skills that are needed to perform all the activities in particular Scrum type. During the Requirements Scrum, the practices *Modernization team* and *Product Owner* address three of the problems by the case study providers presented in the previous section, namely identification of new requirements, specification of cloud-related components and definition of validation criteria for the new requirements. The third issue - evaluation of the requirements, is addressed by the *Product backlog* and *Sprints* practices.

The CS3 answered that Sprints are not fully applicable during the RcS since they needed to perform refactoring of the legacy code, which should be done on the whole code before start of the knowledge acquisition. However, the iterative and incremental cycle of refactoring and code recovery is possible followed by a number of knowledge refinement sprints. Modernization team, including a tool provider as part

of the team, addresses the lack of knowledge problem identified by case studies. CS1 answered that the *Customer reviews* practice is not applicable in their case because the generated models are too finely-grained and it is not possible to be understood by a business domain expert. The loss of data and functionality problem reported by CS2 could be addressed by the *Modelling by two* practice in a business analyst and a developer work together and manually refine the recovered knowledge. *Modernization team* practice addresses the need of interdisciplinary knowledge during the MgS. Emerging requirements during the MgS are handled by one or more Requirements sprints, followed by a number of Migration sprints.

Table 1. Applicability of agile practices. RqS stands for Requirements Scrum; RcS stands for Recovery Scrum; MgS stands for Migration Scrum; '+' sign denotes applicability; '-' sign denotes inapplicability; '/' marks that the practice is not relevant for a scrum.

Agile practice	CS1		CS2			CS3		
	RqS	RcS	RqS	RcS	MgS	RqS	RcS	MgS
Product Owner	+	+	+	+	+	+	+	+
Modernization Team	+	+	+	+	+	+	+	+
Sprints	+	+	+	+	+	+	-	+
Product Backlog	+	+	+	+	+	+	+	+
Modelling by Two	/	-	/	+	/	/	+	/
Customer reviews	+	-	+	+	+	+	+	+
Collective Model Ownership	+	+	/	+	+	+	+	+
Pair Programming	/	/	/	/	+	/	/	-
Collective Code Ownership	/	/	/	/	+	/	/	+

Based on the answers of the questionnaire and the requirements of each case study with respect to the methodology, a customized version of the agile methodology for each case study called Migration paths were created. The migration path for the CS3 is presented in the following section.

5 Agile Methodology Deployment

The section describes an approach to evaluate the deployment of the Agile REMICS Methodology in one case study. The approach suggests a life cycle customised for the particular case study and its execution for part of the functionality of the legacy system. Based on the responses of the questionnaire of agile practices applicability and further interviews with the providers of the CS3, a customized agile methodology was specified and deployed in the project. One team executes the scrums in the project and thus the sprints are sequential. The project starts with initiation project activity during which the system is presented to the team, the goal of migration are discussed and feasibility analysis is performed. First to execute is a RqS with one or two sprints to identify the overall requirements for the new system (i.e. new functional modules, new interfaces and new non-functional requirements). Then, a RcS with two types of sprints is executed. The first type is a refactoring sprint during which the code of the legacy system is refactored to provide proper input for the recovery transformation tool. After the program and database code is successfully recovered a series of refinement sprints are executed. During a refinement sprint the recovered system

knowledge is further reviewed, refined and validated. As soon as a part of the system knowledge is ready to be migrated a MgS starts. Both the new functionality as well as the recovered one is implemented during the sprints inside the MgS. The migrated functionalities are integrated and validated in a VdS. Testing and validation of the integrated part of the system is performed in a sprint or two based on the items in the product backlog for the VdS. Depending on the release strategy, the part of the system might be deployed and a SpS starts. At least one sprint of the SpS is executed to monitor the system execution. Then, the process proceeds to a new requirements, recovery or migration sprint.

The deployment of the agile methodology was done for a part of the legacy system handling maintenance operations. The sprints length is set to two weeks. There is a separate migration team for the VdS for this particular deployment. Since the system has undergone migration following the traditional methodology, some of the activities have been already performed. The overall requirements specification has been done. As well, the code has been refactored and recovered by a recovery tool. Given that, the deployment of the method started with a recovery sprint as part of the RcS. During the RcS the system knowledge regarding planned and unplanned maintenance operations was refined and validated. Then the MgS started with two sprints implementing the functionality recovered in the recovery sprint. The VdS is executed by a dedicated team, which is currently executing the first sprint. In parallel, the other team proceeds to another requirements sprint for the vehicles part inventory functionality, which was decided not to be migrated. The preliminary plan for the pilot deployment is to undergo two other migration sprints, two validation and one supervision sprint.

6 Conclusions

This paper has presented the applicability of agile practices for solving particular problems of model-driven modernization processes by conducting a questionnaire survey on agile practices applicability in particular case studies and by applying an agile modernization methodology in a case study setting.

The contributions of the paper could be summarized as follows: provides description of challenges and problems in traditional model-driven modernization process in three case studies; examines the applicability of particular agile practices for addressing the problems faced by the case study providers; exemplifies the deployment of an agile methodology for model-driven modernization in a project setting.

The findings from the application of the initial methodology in the case studies are currently being used to update and improve the agile version of methodology. Guidelines on possible agile practices that could be applied in the different phases and activity areas are being revised and a new delivery process has been defined according to the presented Modernization Scrum types. We are now preparing a new evaluation round with the case studies to investigate the application of the agile version of the methodology.

Acknowledgements

The research leading to these results has received funding from the European Union Seventh Framework Programme FP7/2007-2013 under grant agreement n° 257793 REMICS. The REMICS Methodology is available online at http://epf.remics.eu/wikis/remics/index.htm

References

1. Comella-Dorda, S., Wallnau, K., Seacord, R. C., & Robert, J. (2000). A Survey of Legacy System Modernization Approaches (Technical Note No. CMU/SEI-2000-TN-003): Carnegie Mellon Software Engineering Institute.
2. Khusidman, V., & Ulrich, W. (2007). Architecture-Driven Modernization: Transforming the Enterprise (White paper No. admtf/07-12-01): Object Management Group (OMG).
3. MOdel driven MOdernization of Complex Systems. (2006-2008) http://www.momocs.org/
4. Matinnejad R., Agile Model Driven Development: An Intelligent Compromise, in Software Engineering Research, Management and Applications (SERA), 2011 9th International Conference on, 2011, pp. 197-202.
5. Picek R., Suitability of Modern Software Development Methodologies for Model Driven Development, Journal of Information and Organizational Sciences, vol. 33, pp. 285-295, 2009.
6. Mahé V., et al., Crossing Model Driven Engineering and Agility: Preliminary Thought on Benefits and Challenges, in 3rd Workshop on Model-Driven Tool & Process Integration, in conjunction with Sxth European Conference on Modelling Foundations and Applications 2010, Paris, France, 2010, pp. 97-108.
7. Christou I., et al., Using the Agile Unified Process in Banking, IEEE Softw., vol. 27, pp. 72-79, 2010.
8. Chung S., et al., A Model-Driven Scrum Process for Service-Oriented Software Reengineering: mScrum4SOSR, in The 2nd International Conference on Computer Science and its Applications (CSA 2009), Jeju Island, Korea, 2009, pp. 1-8.
9. Wang B., et al., A SOA based Model driven Rapid Development Architecture - SMRDA, in The 2nd International Conference on Education Technology and Computer (ICETC 2010), Shanghai, China, 2010.
10. Krasteva I., Stavros S., Ilieva S., Agile Model-Driven Modernization to the Service Cloud, in The Eighth International Conference on Internet and Web Applications and Services ICIW 2013, Rome, Italy
11. REMICS. 2011, Project deliverable: REMICS methodology with agile extension. Available: http://www.remics.eu/publicdeliverables
12. Leire Orue-Echevarria, Marisa Escalante, Juncal Alonso, Gorka Benguria. “Moving to SaaS: Building a Migration Strategy from Concept to Deployment”. DOI:10.4018/978-1-4666-2488-7.ch008.
13. K. Schwaber and J. Sutherland, "The Scrum Guide", Scrum.org, October 2011. http://www.scrum.org/storage/scrumguides/Scrum_Guide.pdf