Agile Model-Driven Modernization to the Service Cloud

Iva Krasteva
Rila Solutions EAD
Acad. G. Bonchev str., bl. 27
Sofia, Bulgaria
ivak@rila.bg

Stavros Stavru
Faculty of Mathematics and Informatics, Sofia University
5, James Boucher Blvd
Sofia, Bulgaria
stavross@fmi.uni-sofia.bg

Sylvia Ilieva
IICT-BAS
Acad. G. Bonchev str., bl. 25A
Sofia, Bulgaria
Sylvia@acad.bg

***Abstract*— Migration of legacy systems to more advanced technologies and platforms is a current issue for many software organizations. Model-Driven Modernization combined with Software as a Service delivery model is a very promising approach, which possesses a lot of advantages, including reduced costs, automation of migration activities and reuse of system functionality. However, a drawback of such an innovative modernization approach is that it lacks mature software process models to guide its adoption. Thus, a methodology for seamless execution of different migration and deployment activities is quite needed. On the other hand, agile development methods have been successfully adopted in various projects, which partly or thoroughly use the engineering and delivery models exploited in the modernization process. This paper presents how a particular methodology for Model-Driven Modernization with deployment to the Cloud is enriched with agile techniques to address different challenging issues. The extended agile methodology could be used by organizations which have already applied agile software development as well as by organizations that plan to introduce it in their work.**

Keywords- Cloud computing; Agile Methodology; Model-driven Modernization; Software as a service

I. INTRODUCTION

Cloud computing and Service-Oriented Architecture (SOA) have recently been recognized as very promising approaches which provide cost-efficient and reliable services. Migration to the Service Cloud paradigm implies transformation of legacy software systems to SOA with deployment in the Cloud. Nowadays, the popularity of the Software as a Service (SaaS) cloud model is growing fast. The SaaS model reduces the infrastructure costs for customers and offers flexible license payment schemas. Despite its numerous advantages, building a SaaS system from scratch to replace the outdated software of an organization might not be a reasonable investment. A modernization approach based on reusing and integrating the company’s legacy applications is a better solution.

Model Driven Modernization is a recent approach, whose aim is to provide automation of most of the migration activities and reuse of legacy functionality. OMG’s Architecture Driven Modernization (ADM) provides support for MDM but is in its earliest stage. They envision a set of automated tools that can disassemble a legacy software system, transform the components in high-level models, reconfigure these models using the best-practices from Model Driven Architecture (MDA), and finally regenerate a modern system.

REMICS (REuse and Migration of legacy systems to Interoperable Cloud Services) is an EU FP7 research project with the objective of supporting the modernization of legacy systems to service cloud by providing a model-driven methodology and tools. REMICS proposes to improve existing approaches and extend them when needed to provide a holistic view of software migration that covers the whole process with a methodology, tools, languages and transformations. The methodology developed in the REMICS project covers the whole life cycle of the migration process. It proposes a traditional sequential approach to software engineering.

Agile methodologies, on the other hand, have been su

ccessfully applied in various contexts, including the ones related to the REMICS project. Thus, the question of whether agile software development can benefit the modernization process is quite adequate and particularly interesting. The aim of the study presented in the paper is to propose and describe a particular agile extension of the general REMICS methodology. A 5-step approach is used to specify how the traditional methodology can be enriched with appropriate agile techniques. The new agile methodology could be used by organizations, which have already applied agile software development as well as by organizations that wish to introduce agile methods in their work.

The rest of the paper is organized as follows. Section 2 presents in more details the REMICS project and related technologies, and thus describes the context of the modernization methodology. A current state-of-the-art of agile adoption in areas related to REMICS is also included. Information on the general REMICS methodology is provided in Section 3. Section 4 describes the approach followed in the creation of the agile extension. In section 5, the scrum types that are defined in the new agile modernization methodology are presented. Section 6 briefly describes how the applicability of the proposed agile REMICS methodology was studied. Finally, Section 7 concludes the paper.

II. BACKGROUND

The present section covers the background on which the agile extension for a migration methodology is created. On one hand, it is the REMICS project, which outlines the specific context of the modernization approach. On the other hand, it is the available research on agile adoption in areas related to this particular modernization approach.

A. *The REMICS Project*

The REMICS project promotes a new development paradigm for migration of legacy systems to the service cloud platforms through innovative model-driven technologies. The model-driven approach followed in the project is taking advantage of OMG's ADM (Architecture-Driven Modernization [1]), KDM (Knowledge Discovery Metamodel [2]), SoaML (Service-oriented architecture Modeling Language [3]) and UML profiles. The baseline concept is the ADM by OMG. In this concept the modernization starts with the extraction of the architecture of the legacy application. Having this architectural model facilitates the analysis of the source system, the identification of the best ways for its modernization and the incorporation of MDE technologies for generating the target system. The project intends to significantly enhance this generic process by proposing a set of advanced technologies for architecture recovery and migration, including innovative technologies such as Model-Driven Interoperability and Models@Runtime. Model-Driven Interoperability is a rather new domain, which builds on top of long history on data and service interoperability. Semi-automated methods that assist users to handle interoperability issues between services are also addressed in REMICS.

The REMICS project includes extending KDM and SoaML to cover concepts related to the migration to SOA and Cloud computing paradigms. Software as a Service (SaaS) is one of the delivery models of Cloud computing whose popularity and usage is growing rapidly in the recent years. The SaaS cloud model provides advantages for both software providers and users. The SaaS delivery model uses a multitenant architecture where a single application is delivered to millions of users through Internet browsers. The advantages offered for the SaaS end user is that installing software is avoided and complex software and hardware requirements can be rapidly fulfilled. In addition, end users do not require upfront licensing and can choose among different payment schemas. Organizations that offer software minimize their support costs and initial investments by outsourcing hardware and software provision and maintenance to the SaaS provider. The provider of SaaS software takes care of the security, availability and performance of the software because they are in charge of the deployment of the system.

A drawback of such a Modernization approach that uses so many innovative technologies is that it lacks mature software process models to guide their adoption. Thus, a methodology for seamless integration and execution of different migration and deployment activities is quite needed. A report on the state of the art on Service Cloud migration methodologies [4] showed that while there are several methodologies for developing service-oriented systems, they are mostly based on developing systems from scratch and not using a legacy system as basis for identifying and implementing services. On the other side, in the context of REMICS project, migration tools and methods need to be integrated with model-based development methods. The migration to interoperable cloud infrastructure introduces some challenges that are not addressed in the current methodologies in a complete way. The state of the art report showed that for the existing cloud computing platforms and cloud deployment model there is no methodology that guides developers through the process of selecting technology and migration to cloud. Additionally, while state of the art in SOA is quite established and covered in literature, the cloud design patterns are more an ad-hoc knowledge that still has to be studied. Existing approaches and methods for transforming a legacy system into a cloud compatible system have some shortcomings in the way they treat interoperability, reliability, scalability, configurability or multi-tenancy. Therefore, a comprehensive process model is needed, which helps organizations improve their technical know-how required for successful migration of their software.

B. *Agile Adoption*

Agile development has been on the cutting edge of both software industry and research for a decade. By shortening time-to-market and responding to the changing requirements of the business, agile approaches are reasonable alternative to the traditional waterfall development methods. The two most popular and widely adopted agile methods through the development community are Scrum [7] and Extreme Programming (XP) [8]. While Scrum provides a framework for managing and organizing agile projects, XP includes practices that are more technologically oriented and supports different development activities such as programming, code integration and testing. The two methods, as well as a hybrid between them, are used in more than two thirds of the agile projects surveyed by VersionOne in 2011 [9].

Nowadays, agile methods and techniques are being widely adapted and successfully applied in many business and application domains, where they are combined with a variety of technologies and platforms, including Model-Driven Development (MDD), Model-Driven Modernization (MDM), SOA and Cloud computing. The combination of MDD and agile software development (known as Agile Model-Driven Development (AMDD)), is the most broadly researched and used among the four technologies mentioned above. A review of existing AMDD methodologies is available by Matinnejad [10], Picek [11] and Mahé et al. [12]. Although there are many existing SOA methodologies, only few methodologies were found to be specifically concerned with incorporating agile software development. Some to mention here are:

- the Agile Service-Oriented Process (ASOP) presented in [13];
- the Xplus framework process model proposed by Shin and Kim [14];

- the combination of Rational Unified Process (RUP) to exploit SOA and a detailed development life cycle based on Agile Unified Process (AUP) [15];
- and the Continuous SCRUM [16].

The later uses a triple-sprint-overlap pattern and some additional best engineering practices in order to sustain a weekly release cycle of a SaaS and PaaS based software system. There are few methodologies, which combine agile software development with both MDD and SOA. The mScrum4SOSR is one such methodology, proposed by Chung et al [17]. The methodology extends Scrum with UML modeling and XP techniques in order to provide comprehensive service-oriented software reengineering process model. Another similar approach is the Model-driven Rapid Development Architecture (SMRDA) - an iterative development approach which unifies SOA and MDD in order to enhance the efficiency of the development efforts and the reusability of the developed services [18].

Based on the conducted review we could claim that there is no agile methodology in the literature which is specifically concerned with model-driven modernization with deployment on the service cloud. However, agile methods as Scrum and XP have been examined, combined and successfully applied in areas, closely related to the REMICS project.

III. General REMICS Migration Methodology

The migration of software systems to cloud infrastructures can be viewed from two different perspectives - business and technology. The business dimension is focussed on the migration of the system to support the cloud business models and usually involves additional functional requirements in order to provide implementation of cloud related support activities such as billing and legalisation. The technology dimension is focussed on the migration of the system so it is able to run in the new cloud environment, taking advantage of the new technologies without adding too much additional functionality. It usually involves new non-functional requirements over the legacy system, as well as few functional requirements.

In general, REMICS migration methodology is focused mainly in the evolution of the technology model. There are 7 activity areas defined in the REMICS methodology, which cover the full life cycle of a legacy system Modernization to the Cloud. Fig. 1 depicts the main activity areas. Tools and techniques in REMICS project support all but the one of the activity areas. Only the withdrawal does not receive special support. For the purpose of the study presented in this paper, a brief description of the activity areas with the composing activities is presented in the sections below. They are the basis on which agile extension will be specified. More detailed information of the methodology is available in [19].

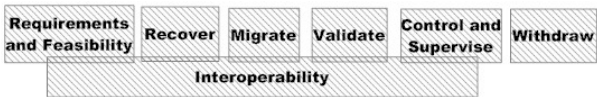

Figure 1. REMICS activity areas

A. *Requirements and Feasibility Activity Area*

In the *requirements and feasibility activity area* the migration requirements for the system are gathered and the main components of the solution and their implementation strategy are identified. The purpose is not an exhaustive description of all requirements of the source system, but the description of the requirements that will require development effort and will be used as a basis for the validation of the system. Feasibility analysis is needed since in a migrated system not all the parts are equally reusable. In some cases the best way to reuse a component may be to wrap it, in other cases to reengineer, or to replace it with an external one, or to implement it from scratch, etc.

The main activities that are included in the *requirements and feasibility activity area* are the following:

- Describe the system;
- Apply techniques to evaluate feasibility;
- Identify actors;
- Identify additional requirements and specify new requirements in UML diagrams;
- Establish validation criteria; and
- Elaborate glossary.

B. *Recover Activity Area*

The purpose of the *recover activity area* is the recovery of the knowledge from those legacy components that during the feasibility analysis has been pointed as candidates to be reengineered. The use of recover methods and tools will provide the application model of the legacy system as well as information on the requirements and the testing procedures for the migrated code. The system knowledge is recovered in KDM format from which UML system models and requirements specification in Requirement Specification Language (RSL) are generated. The following activities are part of the *recover activity area*:

- Collect the code;
- Recover system knowledge;
- Refine system knowledge;
- Generate system model;
- Generate system requirement; and
- Recover application testing.

C. *Migrate Activity Area*

In the *migrate activity area* the target system is defined and implemented using the elements identified during the requirement and recover phases. This includes also the design and implementation of the components necessary for the SaaS application and the development of the service-oriented architecture. The component SOA model is used to define the deployment model of the system. The deployment model contains the identification of the location of the different components distributed in the cloud. When defining this deployment model, it is also necessary to take into account possible constrains (organizational, legal, etc).

The activities that are included in *the migrate activity area* are:

- Complete definition of system requirement;
- Definition of service architecture;

- Definition of cloud architecture;
- Implementation design;
- Generate code stubs of the system; and
- Complete the code.

D. Validation Activity Area

The purpose of the *validation activity area* is to define testing strategy to verify that the migrated system implements the requirements identified and that the components (including those not reengineered) and services work properly. This validation phase includes not only functional validation but what is more important, non-functional validation, especially performance, reliability and security. In the case of cloud computing applications these three aspects must be stressed on. Testing procedures are defined depending on the specifics of the application and particular system requirements.

The following activities are part of the *validation activity area*:

- Define testing infrastructure;
- Identify and refine requirements to be tested;
- Generate acceptance testing;
- Import models elements to be tested;
- Define testing procedures; and
- Implement testing strategy.

E. Supervise Activity Area

The *supervise activity area* provides elements to monitor and control the performance of the system when deployed in the Cloud and to modify that performance. A company can monitor constantly the performance of the application once it has been provisioned as a service, so it can be improved in performance, reliability and resources used. As well, the system can be supervised of possible degradations.

Activities included in the supervision process are:

- Identify monitoring procedures;
- Implement monitoring procedures;
- Monitor the performance of the system in the cloud;
- Detect deviations;
- Perform corrective actions; and
- Monitor corrective actions.

F. Interoperability Activity Area

The *interoperability activity area* provides tools that solve interoperability problems with 3rd part providers or any external components and services. Interoperability is a crosscutting activity to the general methodology that deals with the interoperability issues that affect SaaS along the other activity areas.

Activities for performing interoperability are the following:

- Identify interoperability problems/scenarios;
- Define interoperability related requirements;
- Perform interoperability analysis;
- Implement interoperability components ; and
- Interoperability monitoring.

IV. AGILE EXTENSION TO THE MIGRATION METHODOLOGY

The extension of the general REMICS migration methodology with agile techniques follows a 7-steps approach – identify, analyze, select, define, formalize, evaluate and adapt. The first 5 steps, in which an initial agile methodology is defined, are described in details in the following subsections. The applicability of the initial methodology is studied in the evaluation step and is presented in the following section. As a last step of the approach, further enhancements of the initial methodology are suggested based on the output of the evaluation step. The adaptation step is subject to future research.

A. Identify and Analyze

The identification and analysis of agile methods and techniques in the context of our study was conducted in two consecutive phases. During the first phase, the challenges of all related fields, incl. MDD, SOA, Cloud computing and Software modernization, were extracted, analyzed and synthesized through a systematic literature review [20]. In the second phase various agile techniques were evaluated through the Delphi method in terms of their potential to overcome the extracted challenges.

The systematic review covered total of 84 articles, which were either describing the current state of research and practice in any of the above mentioned four related fields or were identifying and discussing the challenges these areas possess to both academia and industry. The full texts of these articles were thoroughly examined in order to extract all relevant challenges. The extracted challenges were further consolidated into total of 52 challenges and sorted into two categories – organizational and technical challenges. Organizational challenges included process-oriented and people-oriented challenges from all levels of the organization (e.g. competence acquisition, process reengineering, addressing external dependencies, etc.), while technical challenges included design, implementation, verification and deployment challenges, and thus were mostly product and technology-oriented. Among the most cited organizational challenges in the MDD field are the lack of mature tools, integrated development environments and off-the-shelf infrastructure, competence acquisition and reliance on high level models. Popular technical challenges for SOA applications are service design, addressing security, interoperability and other quality aspects and testing services. In the Cloud computing field, among the most cited challenges are trust, security and privacy, external dependencies and vendor lock-ins. Common technical challenges faced during software modernization are ensuring behavioral equivalence and extracting business and technical knowledge from legacy systems. A thorough analysis of the extracted challenges and their implication to agile software development could be found in our previous works [21-22].

During the second phase, various agile techniques were evaluated for their potential to address the challenges, extracted by the review process. These agile techniques were taken from Scrum and XP agile development methods. The methodology used to evaluate these techniques was the Delphi method. More specifically, the Pfeiffer's three step process was followed [23]. During the recommendation step, a panel of experts (with an average of 9 years of both academic and industrial experience in agile software development) was asked to review the list of extracted challenges and recommend agile techniques based on the challenges they could address. In the evaluation step, a consolidated list of recommended agile techniques was sent to each expert to further evaluate the relevance (on a five-point rating scale) of all techniques in regard to all extracted challenges. Finally, during the consensus phase, the consolidated list, together with the experts' ratings was sent once again in order to discuss big differences in ratings and gain consensus. In result, a sorted list of recommended agile techniques was created based on the number of challenges they could address. This list is shown in Table 1.

As seen from Table 1, the agile techniques with the highest rating were Small releases, Planning game and Whole team from XP, and Sprints, Cross-functional teams and Sprint planning meeting from Scrum. Among the arguments for this were receiving feedback quickly, increasing responsiveness to change, building trust and confidence, enhanced collaboration, clarification of team responsibilities and service ownership, early escalation of quality concerns, effective acquisition of competencies and expertise, early delivery of customer value, and many more [21-22]. Other agile techniques, which were also highly recommended by the experts, were Pair programming and Continuous integration from XP, and Product and Sprint backlogs.

TABLE I. EVALUATION OF AGILE TECHNIQUES BASED ON THE CHALLENGES THEY ARE EXPECTED TO ADDRESS

Agile technique	**Number of addressed challenges**
Extreme programming (XP)	
Small releases	38
Planning game	29
Whole team	29
Pair programming	26
Continuous integration	23
Test-driven development	21
System metaphor	14
Collective code ownership	14
Refactoring	9
Simple design	7
Coding standards	6
Scrum	
Sprint	38
Cross-functional teams	35
Sprint planning meeting	33
Product backlog	26
Sprint backlog	26
Product owner	17
Daily scrum	16
Scrum master	13
Scrum of scrums	11
Sprint review meeting	11
Sprint retrospective	7
Sprint burn down chart	4

B. Select

During the selection step, a set of techniques to be included in the initial methodology is identified. As well, techniques that are a subject to modification are chosen. The selection of the initial set of techniques is guided by the principle of incremental methodology design [24], according to which easy to apply techniques are included in the beginning while more challenging techniques are added iteratively. The other two criteria, which are used to select agile techniques for the initial methodology, are taken from the context of the study. The first one is the rating of the agile technique from the sorted list created in the previous step. Some techniques were selected because they were suggested to improve the general REMICS methodology by the industry partners who have been involved in different project case studies. As a result, fourteen techniques out of twenty three are included in the initial agile methodology.

C. Define

During the define step, the new agile methodology is specified. As well, the techniques which are modified to suit the requirements of the migration process and the new ones are described. The agile REMICS methodology proposes a particular implementation of Scrum methodology for large projects. It is based on the so-called Type-C SCRUM [25] in which a number of integrated overlapping scrums are executed. The methodology has been adjusted for the characteristics of the REMICS project by defining a number of Modernization Scrum types. For each of these types, the main Scrum techniques have been modified in the following way:

- Modernization sprints – a number of Sprint types with particular activities depending on REMICS activity areas;
- Modernization product backlogs – each sprint type has a corresponding product backlog. The *Product owner* manages different Product backlog types; and
- Modernization teams - different *Scrum team types are* associated with each Sprint type depending on the required skills for particular sprint type. Teams are self-organizing but with a certain degree of specialization (which is in contrast to general scrum teams) due to the diverse and not so common skills needed for activities in the Sprint types. The Scrum team types integrate the *Whole team* technique of XP as well. Teams are (preferably) collocated and a business expert or a customer is part of the team.

In addition to the Scrum types, there are a couple of agile development techniques that have been adjusted for the extensively used model-driven engineering activities in the REMICS project:

- Modeling by two - based on the *pair programming technique* of XP the modeling by two is done by two people with different roles who work simultaneously on one model in order to exchange, analyze and synthesize knowledge in models better, faster and more easily

- Pair modeling - two analysts work together on a model;
- Collective model ownership - created models are continuously integrated in a common code base; and
- Continuous modeling - models are collectively owned by the whole team.

To make the agile REMICS methodology more effective, a new technique called Shifting team member has been added to the set of agile techniques. A team member moves among teams in different sprint types. When a new sprint begins, (s)he shifts to another team where the team member plays the same or similar role.

D. *Formalize*

The methodology is formally specified using the Eclipse Process Framework (EPF). EPF [26] is an open source solution that implements the SPEM modelling language. SPEM (Software process engineering meta-model) [27] is a specification from the OMG that addresses the standardised definition of software development methodologies. Other tools can latterly automatically process such specifications for different purposes. The agile methodology is based on the general REMICS methodology so they are specified in different packages in which activities refer to each other.

In addition to EPF, the agile REMICS methodology will be implemented in the language proposed in the OMG FACESEM RFP by the SEMAT working group [28]. Both options for specification of REMICS methodology will be compared and the end of the project will give the recommendations for using each one.

V. SCRUM TYPES

The section describes the five scrum types of the proposed initial agile REMICS methodology. To a great extend they conform to the activity areas of the general REMICS methodology. The interoperability activities are added to the related scrum types. Fig. 2 presents the basic components of the scrum types. The activities from the activity areas of the initial REMICS methodology are present in each scrum type. They are modified appropriately to introduce different agile techniques e.g. modelling by two technique is applied in refine system knowledge activity during the recovery scrum. A set of new activities to support agile techniques, such as sprint backlogs and sprint retrospective, are added in each scrum type. Activities that don’t support iterative execution in sprints and serve as preconditions for execution of other activities are gathered in the so called initiation or initialization activity. The initiation activity is executed just ones in the beginning of particular scrum, while the initialization activity might be performed several times in a scrum e.g. each time a new component is to be recovered or migrated.

The presentation here outlines only the major activities of the respective sprint types and how the identified agile techniques are applied. A detailed information of the methodology with activity inputs and outputs, modernization team roles and support materials is available in a final REMICS project report [29]. The last subsection discusses possible life cycles of a Modernization project in which the five scrum types are executed.

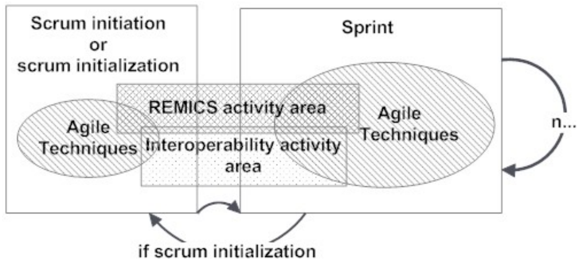

Figure 2. Scrum type

Each project scrum starts with Initiation project activity, which consists of four activities that are performed as sessions and meetings with the whole project team:

- Goals of migration;
- Describe the system;
- Apply techniques to evaluate feasibility;
- Identify actors and initial scrum teams; and
- Prepare the project product backlog and prioritise components for implementation;

The output of the activity is the project product backlog with prioritized components and implementation strategies.

A. *Requirements scrum*

The requirements scrum contains activities for new requirements identification and specification. As well, the scrum handles requirements that address interoperability issues. There are two main activities in the requirements scrum:

- Requirements scrum initiation; and
- Requirements sprints.

The requirements scrum initiation activity is held in the beginning of the scrum and has two activities which are new with regard to the general REMICS methodology:

- Prepare and demo product backlog for requirements scrum; and
- Define general deployment model.

Requirements sprints are one or more sprints executed after the initiation activity. Each requirements sprint starts with a planning meeting and ends with a retrospective. As well, it contains related activities from the general *Requirements and Feasibility activity area* and interoperability requirements identification and specification. The Modelling by two technique is applied for requirements identification and specification when an analyst and a business expert work together on new requirements specification using UML.

B. *Recovery scrum*

The recovery scrum contains activities for recovering of the system requirements from the existing code of the legacy application. There are two main activities in the recovery scrum:

- Recovery initialization; and
- Recovery sprints.

The recovery initialization activity is held every time when the recovery of a particular component starts. The initialization contains three activities:

- Collect the code;
- Recover system knowledge; and
- Prepare and demo product backlog for the component which is to be recovered.

There are a number of recovery sprints that are carried after the recovery initialization. The sprints start with a sprint planning meeting. At the end of the sprints, a retrospective meeting and a demo are performed. The activities in the recovery sprints are executed iteratively and contain refinement of system knowledge, generation of system models and requirements in RSL and recovery of application testing. To support the iterative execution, the technique of continuous modelling and collective model ownership are applied. The system knowledge refinement and system modeling are performed together with a business expert and a developer.

C. *Migration scrum*

The migration scrum contains activities for development of all the system components depending on their implementation strategies – wrapped, recovered and new components. Migration starts when a specification of a component is ready- either of a new component or a recovered one.

There are two main activities in the migration scrum:

- Migration initialization; and
- Migration sprints.

The migration initialization activity is executed in the beginning of each new component that is to be migrated. Migration initialization contains three activities:

- Prepare and demo product backlog for migration scrum;
- Componentization of UML or RSL models;
- Definition of overall cloud architecture.

One or more migration sprints are executed after the initialization activity. Migration sprints contain the activities of the general REMICS migration activity area, interoperability analysis and interoperability components implementation. As during the migration there are a couple of activities, involving knowledge from various innovative and complex areas (e.g. definition of service and cloud architectures), a pairing with a team member technique is suggested. The common for all sprints planning and retrospective activities are also included in the migration sprints. Testing activities of an isolated part of the system is added in the migration sprint so that the demonstrated increment of the system is verified.

D. *Integration and Validation Scrum*

During the integration and validation scrum, the migrated parts of the system are integrated and validation activities of the new functionality are performed. Regression testing is also part of the validation activities. There are two main activities in the integration and validation scrum:

- Integration and validation scrum initialization; and
- Integration and validation sprints.

The initialization activity is performed in the beginning of the integration and validation scrum and each time a part of a new component is migrated. Integration and validation scrum initialization contains three activities:

- Define testing infrastructure;
- Identify and refine requirements to be tested; and
- Create and demo product backlog for the scrum.

After the initialization activity, a number of integration and validation sprints for a particular component are executed. The activities from the general validation activity area are included in the integration and validation sprints. A new integration activity is added since the validation is performed incrementally.

E. *Control and Supervise Scrum*

The control and supervise scrum contains activities for monitoring of the deployed system. It starts when the first release of the system is deployed in the Cloud. There are two main activities in the control and supervise scrum:

- Control and supervise initialization; and
- Control and supervise sprint.

The initialization activity is performed every time a new release is deployed since the monitoring procedures may vary from one release to another. Control and supervise initialization contains two activities:

- Identify monitoring procedures; and
- Prepare the control and supervise product backlog.

After the initialization, there might be one or more sprints to monitor the running system, detect deviations and correct them. When a deviation is identified, it is added to the product backlog to be corrected in some of the following sprints. The control and supervise sprints contain the activities from the corresponding activity area in the general REMICS methodology as well as monitoring of interoperability.

F. *Life Cycle*

A possible life cycle of a modernization project in which the five scrum types are executed is shown on Figure 7. The scrums are overlapping and executed in parallel when possible. The main point is to have releases as soon as possible and to provide continuous and early feedback by a number of sprints. The figure shows one particular lifecycle, however, depending on the project size and staffing possibilities there might be different arrangements of the scrums and sprints. For example, the scrums might be executed successively instead of simultaneously as shown in the figure. As well, the sprints inside each scrum can be executed in parallel, if there are more than one teams of particular type dedicated to the project.

There are forward relations between the scrums as well as backward relations e.g. if during a migration sprint new requirements emerge they will be part of some of the subsequent requirements sprints; problems found during the integration and validation sprint will be handled during the new migration sprints, etc. The project ends with a sprint of the control and supervise scrum.

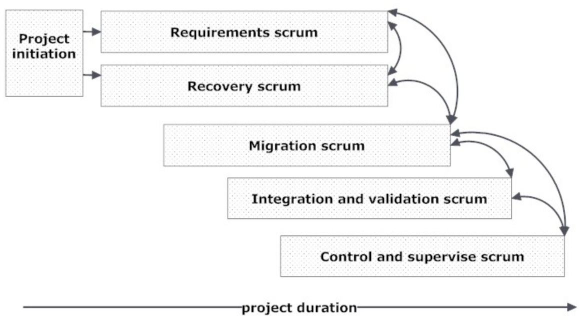

Figure 3. Project life cycle with Modernization scrums

VI. APPLICATION

As part of the evaluation step, the applicability of the proposed agile REMICS methodology was studied in two ways. First, a questionnaire was sent to all of the four case study providers participating in the project. The four case studies differ in business domains and technologies used in their legacy systems. A representative of each case study was asked to answer questions on whether particular agile technique is applicable to his/her case study and to what extent. In addition, for each technique it was studied why it is not applicable and what problems it could address if applied. The information gathered in the questionnaire is supposed to give insight on characteristics of each project as well as specifics of the general modernization process that affect application of the agile REMICS methodology. All the responders has stated moderately or very familiar with both Scrum and XP methods so their judgment could be considered adequate. The overall results showed that most of the suggested agile techniques are applicable in the four case study projects. There were no major issues, considered by the case study providers, which could prevent the proposed agile techniques from applying in their projects.

As a second way for evaluation of the methodology, it was applied in one of the case study projects. Based on the responses of the questionnaire of agile practices applicability and further interviews with the providers of the case study, a customized agile methodology was specified and deployed in the project. Since a pilot functionality of the legacy system was chosen for migration, there was only one team who executed all of the scrums. The agile REMICS methodology was applied for three months with two-week-long sprints. Currently, the results of the application are studied. Satisfaction of the applied techniques is surveyed and suggestions for improvements are gathered. Preliminary results showed that most of the agile techniques were successfully applied, but some adjustments were needed. Based on the feedback provided by case study provider the proposed agile methodology will be further improved.

VII. CONCLUSION

Some of the benefits of using agile methods in software development projects are decreased time-to-market, minimized risk of project failure and growing confidence and satisfaction of project stakeholders. In the present years, the industry is facing urgent need for modernization of outdated software system. Along with that, the popularity of SaaS application is growing fast. The SaaS cloud model provides advantages for both software providers and users. By introducing agile approaches to the development of SaaS applications their numerous advantages could be beneficial for software organizations.

In the current paper, an approach for extension of a particular methodology for model-driven migration of legacy systems to the Service Cloud is proposed. The steps of the suggested approach describe how agile practices and techniques have been identified, analyzed, selected, defined and evaluated to enrich the REMICS project methodology with appropriate techniques. The agile extension of the methodology further addresses challenges of the modernization process and provides support for organizations to move their legacy systems to SaaS applications following the agile development principles. The new methodology is evaluated in industry case studies in two ways. Firstly, a questionnaire of the applicability of agile practices was conducted with the providers of four case studies. Their expert opinion was considered to evaluate applicability of the methodology in different migration project settings. Secondly, the agile REMICS methodology was applied in one case study for three months. Currently, the results of the application are analyzed. Preliminary results showed that most of the agile techniques were successfully applied, but some adjustments need to be done. As a last step of the proposed approach for extension of REMICS modernisation methodology, the proposed agile methodology will be further improved based on the feedback provided by case study provider.

ACKNOWLEDGMENT

The research leading to these results has been developed in the context of the REMICS project (www.remics.eu) partly founded from the European Community's Seventh Framework Programme under grant agreement n° 257793. The authors wish to acknowledge the Commission for their support. We also wish to acknowledge our gratitude and appreciation to all the REMICS Project partners for their contribution during the development of various ideas and concepts presented in this paper.

REFERENCES

[1] ADM. OMG Architecture-Driven Modernization. Available: http://www.omgwiki.org/admtf/doku.php, [retrieved: 05, 2013]

[2] KDM. OMG ADM Knowledge Discovery Metamodel. Available: http://www.omg.org/spec/KDM/, [retrieved: 05, 2013]

[3] SoaML. OMG Service-Oriented Architecture Modeling Language. Available: http://www.omg.org/spec/SoaML/ 1.0.1/PDF/, [retrieved: 05, 2013]

[4] REMICS Project deliverable: State of the art on modernization methodologies, methods and tools. Available: http://www.remics.eu/system/files/REMICS_D2.1_V1.0_LowResolution.pdf, [retrieved: 05, 2013]

[5] S. Ambler, “Agile Software Development at Scale Balancing Agility and Formalism in Software Engineering.” vol. 5082, B. Meyer, et al., Eds., ed: Springer Berlin / Heidelberg, 2008, pp. 1-12.

[6] S. W. Ambler, The Object Primer: Agile Model-Driven Development with UML 2.0: Cambridge University Press, 2004.

[7] K. Beck, Extreme Programming Explained: Embrace Change, Addison_Wesley Professional, 1999

[8] K. Schwaber and M. Beedle, Agile Software Development with Scrum, Prentice Hall, 2002

[9] VersionOne. (2011, State of Agile Development Survey Results. Available: http://www.versionone.com/ state_of_agile_development_survey/11/, [retrieved: 05, 2013]

[10] R. Matinnejad, “Agile Model Driven Development: An Intelligent Compromise,” in Software Engineering Research, Management and Applications (SERA), 2011 9th International Conference on, 2011, pp. 197-202.

[11] R. Picek, “Suitability of Modern Software Development Methodologies for Model Driven Development,” Journal of Information and Organizational Sciences, vol. 33, pp. 285-295, 2009.

[12] V. Mahé, B. Combemale, and J. Cadavid, “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, pp. 97-108.

[13] A. Qumer and B. Henderson-Sellers, “ASOP: An Agile Service-Oriented Process,” Proc. Software Methodologies, Tools and Techniques 07, 2007, pp. 83-92.

[14] S. W. Shin and Haeng Kon Kim, “A Framework for SOA-Based Application on Agile of Small and Medium Enterprise,” in Computer and Information Science , Roger Lee and H. Kim. Eds., Springer Berlin Heidelberg, 2008, pp. 107-120

[15] I. Christou, S. Ponis and E. Palaiologou, “Using the Agile Unified Process in Banking,” IEEE Softw., vol. 27, 2010, pp. 72-79.

[16] P. Agarwal, “Continuous SCRUM: agile management of SAAS products,” Proc. of the 4th India Software Engineering Conference, Thiruvananthapuram, Kerala, India, 2011, pp. 51-60

[17] S. Chung, D. Won, S. Baeg and S. Park, “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.

[18] B. Wang, C. Wen and J. Sheng, “A SOA based Model driven Rapid Development Architecture - SMRDA,” Proc. of the 2nd International Conference on Education Technology and Computer (ICETC 2010), Shanghai, China, 2010, pp. 421-425

[19] REMICS. (2011, Project deliverable: REMICS Methodology. Available: http://www.remics.eu/system/files/REMICS_D2.2_V1.0.pdf, [retrieved: 05, 2013]

[20] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, “Lessons from applying the systematic literature review process within the software engineering domain,” J. Syst. Softw., vol. 80, 2007, pp. 571-583.

[21] S. Stavru, I. Krasteva and S. Ilieva, “Challenges for Migrating to the Service Cloud Paradigm: An Agile Perspective,” Web Information Systems Engineering – WISE 2011 and 2012 Workshops, Springer Berlin Heidelberg, 2012, pp 77-91.

[22] S. Stavru, I. Krasteva and S. Ilieva., “Challenges of Model-Driven Software Modernization: An Agile Perspective,” Proc. The 1st International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2013), Barcelona, Spain, 2013.

[23] J. Pfeiffer, New look at education: systems analysis in our schools and colleges: Odyssey Press, 1968.

[24] K. Beck and C. Andres, Extreme Programming Explained: Embrace Change (2nd Edition): Addison-Wesley Professional, 2004.

[25] J. Sutherland, “Future of Scrum: Parallel Pipelining of Sprints in Complex Projects,” presented at the Proceedings of the Agile Development Conference, 2005.

[26] EPF. Eclipse Process Framework Project (EPF). Available: http://www.eclipse.org/epf/, [retrieved: 05, 2013]

[27] SPEM. OMG Software Process Engineering Meta-model 2.0. Available: http://www.omg.org/spec/SPEM/2.0/PDF/, [retrieved: 05, 2013]

[28] SEMAT working group, Available: http://semat.org/, [retrieved: 05, 2013]

[29] REMICS. Project deliverable: REMICS Methodology with Agile Extension. Available: http://www.remics.eu/ publicdeliverables [retrieved: 05, 2013]