Reuse and Migration of Legacy Systems to Interoperable Cloud Services- The REMICS project

Parastoo Mohagheghi[1], Arne J. Berre[1], Andrey Sadovykh[2], Franck Barbier[3], Gorka Benguria[4]
[1]SINTEF, Forskningsveien 1, 0373 Oslo- Norway
[2]SOFTEAM, 21 avenue Victor Hugo, 75016 Paris- France
[3]Univ. of Pau - Netfective Technology- France
[4]Fundación European Software Institute, Parque Tecnologico #204, 48170 Zamudio- Spain
{parastoo.mohagheghi, arne.j.berre}@sintef.no, andrey.sadovykh@softeam.fr, franckbarbier@franckbarbier.com, gorka.benguria@esi.es

Abstract

Legacy applications are sometimes of substantial value for companies when moving towards the future internet paradigm. Despite the fact that technologies of legacy systems may be outdated, they still function for the users' needs, capture important business logic and the cost of replacing them with systems designed from scratch is often too high. The objective of the REMICS project is providing tools and methods to support the migration of legacy systems into the service cloud paradigm based on a Service-Oriented Architecture (SOA). The migration process consists of understanding the legacy system in terms of its architecture and functions, designing a new SOA application that provides the same or better functionality, and verifying and implementing the new application in the cloud. REMICS proposes a methodology that covers these steps based on model-driven engineering tools and methods.

1. Introduction

Cloud computing and Service-Oriented Architecture (SOA) are recognized as game-changing technologies for a cost-efficient and reliable service delivery [6]. The Software as a Service (SaaS) paradigm becomes more and more popular enabling flexible license payment schemas and moving the infrastructure management costs from consumers to service providers. However, building a SaaS system from scratch may require a huge investment in time and efforts. Therefore, it is necessary to build the system by reusing and integrating the legacy applications of an organisation. Unfortunately, this is not an easy task and organizations legacy systems are often difficult to reuse due to platform, documentation and architecture obsolescence.

REMICS (REuse and Migration of legacy systems to Interoperable Cloud Services) is a research project (STREP) accepted in the Objective 1.2 of FP7 Call 5 (Internet of Services, Software and virtualization) with the objective of supporting the migration of legacy systems to service clouds by providing a model-driven methodology and tools.

Model-Driven Engineering (MDE) technologies such as the OMG's MDA [5] (Model-Driven Architecture) put models in the centre of the software engineering process. The software products are built with subsequent model refinements and transformations from business models (process, rules, motivation), down to component architectures (e.g. SOA), detailed platform specific design and finally implementation. Similarly, OMG's ADM [3] (Architecture-Driven Modernization) proposes to start with knowledge discovery from legacy systems in order to recover models, define the target architecture and then transform the system from the as-is state to the to-be state. REMICS proposes to improve existing approaches and extend them when needed to provide a holistic view to migration that covers the whole process with a methodology, tools, languages and transformations. One main objective of REMICS is to provide its solutions based on standards and open source tools as much as possible to facilitate reuse and shorten the barrier for users to take advantage of the innovations.

The remainder of this paper is organized as follows. Section 2 is a short discussion of challenges in the migration process based on the state-of-the-art analysis and experiences of the two industrial pilot cases in the

REMICS project. Section 3 presents the vision of REMICS and how the challenges will be addressed. Section 4 is a short introduction of the contributions related to standards and Section 5 is conclusions.

2. Research challenges

The need for agility for IT infrastructures and information systems forces organizations to move to up-to-date technologies and applications. Years after years, the explosion of costs generated by the maintenance of old systems, also becomes a critical issue. The huge heap of data, code and tailored architectures organizing these data and code, is a great barrier to modernize legacy systems. Even though MDE is nowadays recognized as a suitable technical framework to tackle this problem, one may list recurrent challenges to be addressed:

- *The oldness degree of the technologies to be reversed; the use of eclectic technologies with non-common gateways and the mixing of execution environments.* Legacy systems often reveal very specific software architectures with homemade tuning. For example in REMICS, case studies have parts with heterogeneous technologies like COBOL[1] on one side, Delphi on another side and even .NET/C# and PL/SQL parts that require very specific platforms to work. Moreover, old software practices weakly separated data formats, code architectural features and infrastructures (e.g., CICS for COBOL). This absence of separation is one the greater challenge when extracting and thus organizing the old material in distinct concerns[2].
- *The absence of synthetic knowledge, and the sharing of knowledge between previous and current employees.* Knowledge on legacy systems covers knowledge relating to all technical facets of the existing IT solution. More critically, this also relates to the business knowledge that is in-depth graved in legacy information systems. The extraction of business rules and more generally business intelligence is a key challenge. The move from one technical solution to a more efficient one, supposes that, *a minima*, the migration process favours functionality-equivalent applications, and great flexibility for unanticipated extensions as well. REMICS' case studies in the tourism, accounting and CRM (Customer Relationship Management) domains show the need to maintaining, even increasing the business advantage of the companies holding these IT solutions.
- *In the same line of reasoning, modernized systems must have at least equivalent QoS (Quality of Service) properties.* The strong optimization of legacy systems resulting from years of continuous effort, is a great difficulty. The distance between the old system in functioning and the planned replacing solution, requires a unified view with measurement possibilities. An appropriate investigation track in REMICS is to extract and/or ask for test data and scenario from the old system with a technology-free expressiveness to be replayed for the new system. More generally, validation is a key issue (see also Section 3) to demonstrate that the new system has at least similar (better) properties compared to the legacy one.
- *One neglected point is the psychological facet along with the cost generated by the migration effort itself.* Only progressive migrations, from small (representative) pieces to larger (more critical) pieces, are tolerable. Convincing and training people in MDE is also a key aspect in the sense that legacy information systems do not promote abstraction at all; the contrary of models! However, the graphical nature of models may play a great role in the acquisition of a global unified view (and thus comprehension) of the old system, even if no generation of a new system is envisaged at short-term.

From the transformation principle at the heart of MDE, one of course expects, in REMICS especially, to address all the above listed issues with successful results. The key forces of MDE are especially:

- The alignment with existing dedicated ADM standards (see also Section 4), namely Knowledge Discovery Metamodel (KDM) [4]. One added value of REMICS is a competitive advantage in experimenting KDM with prior small-scale experiments. One challenge here is the systematic large-scale generalization and use of KDM. Improvements of this standard will lead to amendment proposals towards the OMG ADM task force.

[1] COBOL itself is subject to varied dialects. Supersets also exist based on a richer COBOL often creating stronger architectural dependencies.

[2] One may notice that some extracted pieces are no longer of value and/or of sense in the renewed system. To sort among these pieces is also a key challenge.

- Nowadays, the modularity and scalability of transformations are a reality. The ability to divide transformations in cohesive parts and the possibility to weave these parts to construct complex transformations guarantees a scalable and fairly generic approach in REMICS. The REMICS method and tools intend to be portable by construction.
- An important assumption in REMICS is the ability to push the abstracted system to any open technology with a focus on cloud computing. These are the two extremes of the PIM/PSM[3] precept: legacy information systems are characterized by a direct unchangeable way by which IT resources are used and controlled, including screens for example in COBOL, while MDE and clouds in essence cut as much as possible dependencies with physical infrastructures (Infrastructure-as-a-Service or IaaS) and platforms (Platform-as-a-Service or PaaS). Performance through ease of load balancing and more generally any QoS feature, are better taking into account with virtualization offered by clouds. This imposes to endow PIMs representing legacy systems with cloud-based annotations and associated implementations in PSMs' generation.

Based on the above, the main challenges addressed by the REMICS project include:

- How to efficiently extract business value information from legacy system artefacts?
- How to extend the SOA models with peculiarities of the cloud computing paradigm?
- How to automate the architecture migration with model transformation?
- How to efficiently adapt existing services and increase their interoperability to cope with the migration requirements?
- How to reuse the legacy applications in automated testing of the new migrated SaaS application?
- How to manage the SaaS application at runtime using the available models?

Next section gives some details on the REMICS' technical orientations and innovations to achieve all of these goals.

3. REMICS technological approach

The REMICS project will promote a new development paradigm for migration of existing IT systems to the service cloud platforms through innovative model-driven technologies. 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 the architectural model helps to analyze the legacy system, identify the best ways for modernization and benefit from MDE technologies for generation of the new system.

The project intends to significantly enhance this generic process by proposing sets of advanced technologies for architecture recovery and migration involving the innovative technologies such as Model-Driven Interoperability and Models@Runtime.

Fig. 1 depicts the overall REMICS process. In order to "Recover" the Source Architecture, the process starts with the analysis of the available legacy system artefacts: source code, binaries, documentation, users' knowledge, configuration files, execution logs and traces. This activity will be supported by automated reverse engineering and knowledge discovery methods. The REMICS project targets to extract different kinds of information using the KDM by OMG. The information extracted from the recover phase should support the decision about the better strategy for legacy system migration. This information will be then translated into models covering different aspects of the architecture: Business Process, Business Rules, Components, Implementation, and Test specifications depending on the migration strategy.

These models will be the starting point for the "Migrate" activity. During this activity, the new architecture of the migrated system will be built by applying specific SOA/cloud computing patterns and methods like architecture decomposition, legacy components wrapping and legacy components replacement with new discovered cloud services.

These methods will be complimented with generic "Design by Service Composition" methods providing developers with tools simplifying development by reusing the services and components available in the cloud. The system will be rebuilt for a new platform in a forward MDA process by applying specific transformations dedicated to service cloud platforms.

The migration process will be supported by two complementary activities: "Model-Driven Interoperability" and "Validate, Control and Supervise".

[3] Platform Independent Model vs. Platform Specific Model

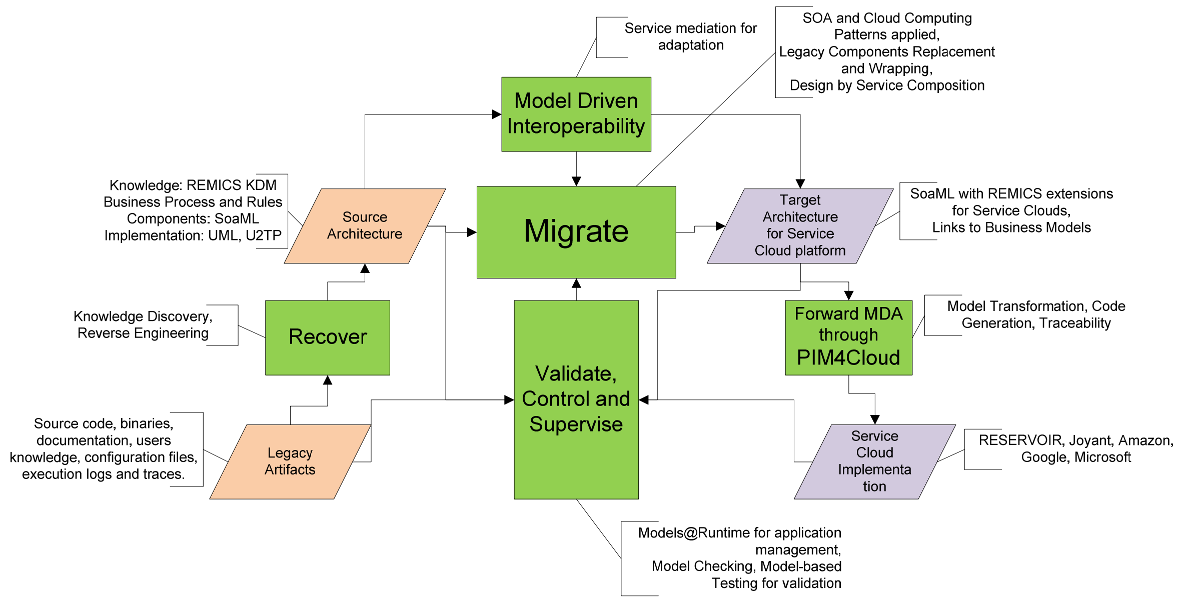
Service mediation for adaptation
SOA and Cloud Computing Patterns applied, Legacy Components Replacement and Wrapping, Design by Service Composition
Model Driven Interoperability
Knowledge: REMICS KDM Business Process and Rules Components: SoaML Implementation: UML, U2TP
Source Architecture
Migrate
Target Architecture for Service Cloud platform
SoaML with REMICS extensions for Service Clouds, Links to Business Models
Knowledge Discovery, Reverse Engineering
Recover
Validate, Control and Supervise
Forward MDA through PIM4Cloud
Model Transformation, Code Generation, Traceability
Source code, binaries, documentation, users knowledge, configuration files, execution logs and traces.
Legacy Artifacts
Service Cloud Implementation
RESERVOIR, Joyant, Amazon, Google, Microsoft
Models@Runtime for application management, Model Checking, Model-based Testing for validation

Fig. 1. REMICS approach

Model-Driven Interoperability (MDI) aims at adapting existing services to services required to complete the Target Architecture in component replacement or service composition sub-activities. The MDI technology will use mediation modelling and automatic model transformations for service interoperability.

The "Validate, Control and Supervise" activity groups the technologies dedicated to ensure the correctness of migrated IT system. Indeed, it should be ensured that, e.g., the recovered Source Architecture correspond to the Legacy System both for structural and behavioural aspects. In the "Migrate" activity, the Business Process and Rules of the Source Architecture have to be fully implemented in Target Architecture. Finally, the system rebuilt for a service cloud platform has to fully address the business goals, process, rules, and QoS. For this a set of model-driven technologies will be applied and adapted for service cloud platforms. Model checking will be applied for the verification of source and target models for various properties: coverage, conformance to design conventions and etc. Model-based Testing will be applied to derive the test scenarios and execute test cases using the legacy system as an oracle. Finally, the Model@Runtime technology will be applied for application management at runtime.

The REMICS project will derive user requirements and validate its results on two industrial user pilots. One pilot is within the ERP/Accounting/CRM domain (DI Systemer from Norway) and the second pilot is within the tourism sector (DOME Consulting and Solutions from Spain). Research partners and tool providers in the project are SINTEF from Norway, SOFTEAM and Netfective Technology from France, Fraunhofer (Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung) from Germany, and ESI (Fundación European Software Institute) from Spain.

In the remainder of this section, we discuss activities in Fig. 1 in more detail.

3.1 Recover

The large-scale adoption of SOA and cloud applications stumbles over the possibility to interconnect clouds with legacy information systems. This interconnection cannot occur without a strong "renewal" of legacy applications: their componentization, their transformation in the form manageable pieces (models in the sense of Model-Driven Development) and their possible redesign.

One of the most important outcomes from the recover phase is the analysis of the feasibility of the modernization of the different parts of the components of the legacy applications. In this task there are some evaluation methodologies that could be used such as SMART (Service Migration and Reuse Technique) [2]. Based on the information collected about the legacy system, the method provides a preliminary feasibility analysis of the viability and the migration strategies. It also provides preliminary estimates about the cost and the risks involved. This methodology is focused on the migration feasibility to SOA environments, not to cloud, therefore some adaptation may be required.

Another outcome will be the migration strategy to be applied to the different components of the legacy system. There are many migration strategies described in the state of the art, for example the “The Legacy Application Modernization Benchmark Report” identifies four possible migration strategies [1]: modernize, extend, surround and replace. Modernize means to update the legacy to new technologies (new database, new programming language, etc). The implication is that the legacy stays in the same place and is changed to be updated. The extend and surround strategies leave the legacy at it is. Extends increases the legacy to include new features (such as web service communication), while surround creates a parallel system in charge of the new features. Finally replace means to discard the legacy and build a new application that duplicates its functionality. Not all these strategies make sense in the REMICS context and proper strategies will be identified during the project.

The recovery process may be based on reverse engineering using standards like OMG KDM or ICT technologies like MOMOCS [7]. This amounts to reversing code in a componentized way. Business rules for instance engraved in legacy systems must be distinguished from any adherence to old technologies like COBOL-like fourth-generation programming languages, mainframe APIs, etc. From an economical viewpoint, all legacy elements cannot be subject to such sizable reverse efforts which imply redeployment to modern platforms.

REMICS will develop practical methods on recovering business value information from the legacy system artefacts based on KDM extensions. These methods will allow recovering the essence of the IT system: business requirements, process, rules, non-functional properties component architecture, semantics, valuable implementation details and test specifications.

The ability to formalize cloud applications including legacy and new services through Platform-Independent Models (PIM) written by means of SoaML extensions, enables us to assess applications independently of deployment platforms. REMICS aims at tackling this problem in an innovative fashion by means of executable models. Models of cloud-based applications which abstractly include legacy and non-legacy will benefit from being simulated thanks to Models@Runtime

3.2 Migrate

The distinguishing peculiarity of service cloud development paradigm is that the refactored legacy architecture has to be decoupled into highly autonomous components, which have to be capable to be dynamically reconfigured. They should have the capability to dynamically augment the number of their instances at runtime using virtualisation features. In some cases the number of component instances required to meet a peak demand is not known in advance. For these cases the application should foresee computational resources request procedures, dynamic resource allocation and be prepared to deal with all issues that may arise. A part of the application should be aware of the current load and resources required in the near future. The migration goal is to re-architecture the legacy systems to a dynamically reconfigurable SOA.

In order to save costs for the service cloud application development, the project proposes to start the migration process from the recovered model, which integrates the essence of the legacy system: business requirements, process, rules, non-functional properties component architecture, semantics, valuable implementation details and test specifications. However this model has to be drastically refactored. The monolithic components have to be decomposed and decoupled in order to obtain an autonomous SOA architecture. The replacement components and on-line services should be discovered, and integrated into design. When the replacement is not found, Model-Driven Interoperability (MDI) may help to adapt the exiting services using automatically generated mediators. Moreover, for components that are chosen for redevelopment, it may be considered to extract a piece of the legacy and wrap it into an SOA packaging.

In order to ensure that the redesigned model fully implements the business requirements, rules and process of the original legacy system, the project proposes to explore the use of Models@Runtime approach. Indeed, using the Models@Runtime principles for simulations would allow ensuring the functional compatibility between the new and the original components. In addition, investing into Models@Runtime during the design phase is viewed as a way to leverage the manageability of applications.

3.3 Interoperate

When migrating to the service cloud one has the possibility to add new functionality to the original system by combining the migrated services in a way they where not combined in the original system. This is one of the strengths of service-oriented architectures. Service composition is a way to achieve this. With the use of model-driven service composition, new services can be created on top of the migrated services and do not need to deal with the underlying technologies. This facilitates developing new services based on existing services because the developer does not need to have an extensive knowledge of the underlying service infrastructure or the legacy applications to be able to develop new services.

One challenge with service composition is the need for mediation, i.e. aligning the data formats of the services in a service composition. Mediation emerges as a necessity when one has services representing data in different formats or you only want a subset of what is offered as result from other services. As a possible solution to this we see mediation services, services that can take input data in one format and outputs the same data only in another format. Creating these mediation services can be done by using model-driven development. This requires a way to create models of how to perform a mediation of a given data format. This can be done by combining a SoaML model with the data format model and a behavioural model for the mediation.

By extending SoaML with mediation modelling we get a PIM4ServiceInteroperability that is also subject of research in REMICS.

3.4 Validate

The recovered solution must be validated against the original requirements. The validation process comprises several steps. The specification of the system which was recovered by using sources like the original specification of the system as well as the system itself has to be validated. Since the projects use a model-based approach, several different model-based validation techniques can be used here, ranging from constraint checking engines to traditional model checking techniques depending on the kind of models that have been recovered. It is important to facilitate the process of verifying and validating the specification with new methodologies, since this activity requires a lot of effort and time. So based on a customizable set of rules, a number of validation and verification processes and goals can be generated individually for a recovered specification. These processes can be executed automatically using several different technologies. The set of rule takes the specifics of the original system as well as of the new cloud paradigm into account. With such a support verification and validation of the recovered system specification is easier and can be applied to a much greater extend. This increases the confidence in the verification and validation result.

In addition to the verification and validation of the specification, the validation of the migrated system is important as well. A model-based testing approach is used here. The test cases to be used in order to validate the system can either come from originally specified ones, which are then recovered in the same approach as the system is recovered, or the test cases are newly derived from the recovered specification taking original user requirements into account. The test cases needed for the migrated system have to serve the purpose of comparing the original system with the migrated one in terms of non-functional properties such as performance, robustness, scalability.

An important aspect to assess the test cases which are executed against the migrated system is that the original system can serve as a test oracle. So the test cases can be run against the old system and the new system in parallel which will allow the comparison of both systems in terms of functionality and in terms of non-functional aspects.

The consideration of the original user requirements is an important challenge, since they are mostly not well captured or not formalized. So a good alignment with the architecture recovery process is mandatory.

3.5 Control and Supervise

Application control and supervision refers to managing applications by means of a programmable management infrastructure. For example, JMX, standing for *Java Management eXtensions*, is a technology standard for Java-based enterprise applications. Moreover, this standard is seamlessly integrated with other Java technologies like Java Web Services for example to instrument and enable the management of such services at runtime. This is equivalent to the notion of managed code in the .NET platform. Management support most of the time rely on metadata and/or reflexive facilities of platforms.

In the REMICS context, virtualisation increases the need for supervision (application behaviour and QoS observations) and control (corrective actions) because changes and adaptations at runtime occur in a more sizable way.

While management technologies and dedicated protocols exist, no method to design and support manageable pieces at runtime exists. In REMICS, if the migrated system is formally specified by using a model-driven approach, keeping Models@Runtime is an appropriate approach for management, even self-management. This is especially facilitated by metamodeling. Specifically, cloud platforms require, compared to traditional SOA, more manageability support, even self-manageability. The ability to transparently scale up, to have fail-in-place components without disturbances illustrates manageability. With clouds, developers must have at their own disposal dedicated APIs. The question in

REMICS is what can these APIs be made up? REMICS view offers actions on Models@Runtime as management functions. Developers may develop self-management functions in their cloud services (self-healing for fault resilience, self-optimization for scalability, etc.)

4. Contributions to standards

Standardization is important to ensure that the technologies developed within a research project are widely accepted and available. Through the REMICS project, we will establish a global strategy in order to promote the REMICS results to standardization bodies. Submissions to Request For Proposals (RFP) will be jointly prepared. For each standardization body, the participation of the members of the REMICS Consortium will be coordinated.

PIM4CLOUD (or CLOUDML) will be proposed for standardization through an OMG RFP process. It will be considered as an extension to the OMG SoaML (Service-oriented architecture Modelling Language) that currently is being standardized based on contributions and results from an international team, including the FP7 SHAPE project (2007-2010) [8], led by SINTEF. SoaML is by end of 2009 in it's last phase of adoption through the Finalization Task Force led by Dr. Arne J. Berre from SINTEF and Jim Amsden from IBM. The OMG SoaML standard provides a UML profile and metamodel for the specification of services. SoaML can be used for architecture level modelling, or as part of a model-driven architecture (MDA) process, starting with a business model and transitioning through logical and physical models, resulting in technology implementation. The focus of PIM4CLOUD will be on describing viewpoint models from different stakeholder perspectives suitable for the cloud computing paradigm. Discussion about the need and usefulness for a platform independent and technology neutral standard for cloud computing has already started, and there is considerable interest in the OMG community for doing this. The initial steps of drafting an RFP will probably happen during 2010, and the initial submissions would then be expected in 2011.

In the new BPMN 2.0 standard draft from OMG there is a model for service modelling that defines collaboration models which also have a relationship to SoaML. We will look into how business process models and service models with a basis in SoaML can be aligned, and propose appropriate updates to both the SoaML standard and the BPMN standard – based on the experiences from using these two standards in the REMICS project.

The Knowledge Discovery Metamodel (KDM) standard represents a general purpose format for storing the information extracted from legacy artefacts, including code, platform, user interface, data, events, deployment and runtime. REMICS will study the ways on linking this knowledge with business models: processes and rules. Based on this experience, the consortium will propose the extensions to KDM required fulfilling the REMICS methodology.

Other extensions to standards for the modelling of SaaS paradigm on the platform independent level will be PIM4ServiceInteroprability, QoS4SoaML, and PIM4Models@Runtime. PIM4ServiceInteroprability will be focused on the support for representing mediators or data model transformations. QoS4SoaML will enable the description of the expected QoS in order to test it during the testing and runtime of the new system. Finally, PIM4Models@Runtime will be focused in the runtime adaptation and management of the resulting system.

Several years ago, Fraunhofer initiated and led the OMG standardization process for the UML2 Testing Profile (U2TP) which introduced dedicated means for testing into UML2. Meanwhile, due to the further evolvement of UML, U2TP needs to be updated. In addition, the project will investigate further refinements and extensions to U2TP needed for the SOA and cloud computing domain. The results will be standardized in OMG.

5. Conclusions

The REMICS project addresses several challenges of the migration of legacy systems to service cloud platforms. Our approach to migration preserves and capitalizes on the business value engraved in legacy systems to gain flexibility brought by service clouds; thus it will lower the cost of service provision and shorten the time-to-market for providing services.

The MDE approach used in REMICS facilitates future evolution of the system along the time. This will have a positive effect in the maintenance costs of the system and will allow the consolidation of organization IT infrastructure.

The REMICS approach described in Section 5 includes five areas that together address the challenges introduced in the Section 2 of this paper:

- Recover will support the efficient extraction of the business value information from the legacy system artefacts.

- Migrate will extend the SOA models with cloud specific elements that using MDA techniques will support architecture migration.
- Interoperate will provide new systems that are adaptable and interoperable as needed.
- Validate will support robust and automated testing strategies that guarantee that the new system provide the same functionality with the same QoS.
- Finally, Control and Supervise will help to manage the SaaS application at runtime using the available models.

For the best exploitation of the results inside and outside of the consortium, the project will focus on standardization, Open source Metamodels, Open Models for standards and Open Interfaces. We will typically provide Eclipse based versions of the models representing the standards provided in REMICS. This will allow easy access to the standards and easy implementation of these by multiple vendors.

REMICS is planned to start in September 2010 and will run for three years. Project results will be available on the project website (http://www.remics.eu) and by publications.

Acknowledgements. This paper is written based on the REMICS project proposal and we acknowledge the contributions of all partners.

6. References

[1] Aberdeen Group, “The Legacy Application Modernization Benchmark Report”, September 2006. http://www.aberdeen.com/Aberdeen-Library/3485/RA_SOLA_PK_3485.aspx

[2] G.A. Lewis, E.J. Morris, D.B. Smith, and S. Simanta “SMART: Analyzing the Reuse Potential of Legacy Components in a Service-Oriented Architecture Environment”, CMU/SEI-2008-TN-008, June 2008.

[3] OMG ADM, http://adm.omg.org/

[4] OMG KDM, http://www.omg.org/technology/kdm/

[5] OMG MDA, http://www.omg.org/mda

[6] Pierre Audoin Consultants Report, “Economic and Social Impact of Software & Software-Based Services, D2 – The European Software Industry”, July 30, 2009. http://cordis.europa.eu/fp7/ict/ssai/study-sw-2009_en.html.

[7] MOMOCS project, http://www.momocs.org/.

[8] SHAPE project, http://www.shape-project.eu/.