Software Engineering Challenges for Migration to the Service Cloud Paradigm

On-going Work in the REMICS Project

Parastoo Mohagheghi
SINTEF ICT
Oslo, Norway
Parastoo.Mohagheghi@sintef.no

Thor Sæther
DI Systemer AS
Trondheim, Norway
Thor.Saether@disystemer.no

***Abstract*—This paper presents on-going work in a research project on defining methodology and tools for model-driven migration of legacy applications to a service-oriented architecture with deployment in the cloud; i.e. the Service Cloud paradigm. We have performed a comprehensive state of the art analysis and present some findings here. In parallel, the two industrial participants in the project have specified their requirements and expectations regarding modernization of their applications. The SOA paradigm implies the breakdown of architecture into high-grain components providing business services. For taking advantage of the services of cloud computing technologies, the clients' architecture should be decomposed, decoupled and be made scalable. Also requirements regarding servers, data storage and security, networking and response time, business models and pricing should be projected. We present software engineering challenges related to these aspects and examples of these in the context of one of the industrial cases in the project.**

Keywords-cloud computing; service-oriented architecture; methodology; software engineering, migration

I. Introduction

REMICS (REuse and Migration of legacy applications to Interoperable Cloud Services [17]) is a research project supported by the European Commission that started in 2010 and will run for three years. The project's main objective is to develop a set of model-driven methods and tools that support organizations with legacy systems to modernize them according to the "Service Cloud paradigm". In our view, the cloud computing paradigm enhances thinking of IT companies as service providers. We therefore talk of the "Service Cloud paradigm" that stands for the combination of cloud computing and Service-Oriented Architecture (SOA) for the development of Software as a Service (SaaS) systems. As stated in [18], one of the most attractive promises of a SOA environment is that it enables reuse of legacy systems, thereby providing a significant return on the investment in these systems. However, migrating legacy systems is neither automatic nor easy.

The first phase of REMICS concentrated on performing a comprehensive state of the art analysis on software methodologies in general, service-oriented methodologies, cloud computing platforms, SOA and cloud design patterns, recovery and migration methods and tools, and verification and validation methods. The results are now published on the project website. In parallel, the two industrial partners in the project have specified their requirements and developed models of their legacy systems. In this paper we present experiences of this phase and discuss challenges and future work, with focus on software engineering challenges.

The remainder of the paper is organized as follows. Section II presents state of the art relevant to our discussion while Section III is an introduction to the REMICS approach for migration. Section IV presents the findings regarding software engineering challenges. Section V is an introduction to an industrial pilot case in REMICS as an example of the challenges of migration. Finally, Section VI contains conclusions and future work.

II. A Brief Overview of the State of the Art

In this paper we assume that the reader has some knowledge of SOA concepts. There are already several service engineering methodologies such as:

- SAE, Service Architecture Engineering, which extends the reference model by OASIS for SOA [4];
- SOAD, Service-Oriented Analysis and Design, developed by IBM [21];
- SODA, Service Oriented Development of Applications, developed by Gartner Research and with emphasize on reuse [7];
- SOMA, Service-Oriented Modeling and Architecture [2], also developed by IBM.

The above methodologies focus on activities in developing systems based on SOA, and some with specific focus on software reuse. For example SOAD includes the activity of *service identification* which consists of a combination of top-down, bottom-up, and middle-out techniques of domain decomposition, existing asset analysis, and goal-service modeling. The same activity is included in SOMA. However SOMA also emphasizes that the design strategy for a SOA is not bottom-up since SOA is more strategic and business-aligned.

SMART (Service Migration and Reuse Technique) developed by Carnegie Mellon® Software Engineering Institute (SEI) [18] is another methodology with focus on modernizing legacy systems to a SOA. It is defined as a process to help organizations to make initial decisions about the feasibility of reusing legacy components as services within a SOA environment. The implementation of SMART involves five major composite activities: establishing stakeholder context, describe existing capability, describe the SOA state, analyze the gap, and develop migration strategy. We found this process especially relevant in our context.

There is extensive publication on the subject of reverse engineering and modernization. Comella-Dorda et al. give a survey of legacy system modernisation approaches [5]. They distinguish between two main types of modernisation: white-box and black-box modernisation. White box modernisation requires an understanding of the internal parts of a system, and involves re-structuring, re-architecting, and re-implementing the system. Black-box modernisation is only concerned with the input/output, i.e. the interfaces, of the legacy system, and is often based on wrapping. Our approach in REMICS involves some re-structuring and re-architecting while parts of the applications will be reused in a black-box way.

Razavian and Lago present a SOA migration framework (SOA-MF) wherein they establish an overall process framework for legacy migration, focusing on recovery and re-engineering, and put it in the context of migration methods such as SMART [16]. In their view, migration is a process of reverse engineering, transformation and forward engineering. Several tiers of artifacts are involved in the migration process; i.e. code, basic design elements, composite design elements, and concepts and business processes.

There is also a number of migration methods developed in various research projects. One of them is the XIRUP process for modernization developed in the MOMOCS project ((MOdel driven MOdernisation of Complex Systems) [10]. The phases of the XIRUP process are preliminary evaluation, understanding, building and migration. The method relies on models and transformations but does not include services and SOA.

SINTEF has extensive experience on model-driven approaches which will be applied in the project. We will also use methods and experiences from the SiSaS project (Scientific Software as a Service) such as a UML profile for migration as described in [11].

We also performed state of the art analysis regarding cloud computing technologies. The notion of "cloud computing" groups together several heterogeneous forms in terms of the services provided and the types of implementation. Regarding provided services we differ between:

- The *Software as a Service (SaaS)* approach, where the service provider makes available software in the form of an internet service. Users access this application, generally paying each time they use it.
- The *Platform as a Service (PaaS)* approach**,** where the service provider makes available a development environment that includes the operating system, as well as a set of services dedicated to the development, testing, deployment and hosting of sophisticated web applications.
- The *Infrastructure as a Service (IaaS)* approach**,** where the service provider makes a resource virtualization platform available as a service. Application developers can access calculation, data storage and network management infrastructures on demand.

There are also different types of cloud, each meeting specific needs. The main types are:

- *Public cloud* in the case a service provider wishes to provide services to public users over the internet;
- *Private cloud* in the case an organization installs its own farm of servers and deploys a cloud computing infrastructure for its exclusive use;
- *Hybrid cloud* in the case an organization wishes to combine a private and public cloud infrastructure to gain more flexibility.

In our state of the art analysis we compared Windows Azure [20] with Google App Engine [8] as PaaS providers and realized that both only support public clouds but with various architectural solutions. Regarding IaaS platforms, we compared the Amazon Elastic Compute Cloud or Amazon EC2 [1] with a few open source technologies such as Eucalyptus [6], OpenNebula [14] and OpenStack [15]. Our experiments on IaaS level platforms have shown that this type of platform offers more flexibility than PaaS-level platforms and also the possibility to have a private, public or hybrid cloud. Additionally, Amazon EC2 acts as a de-facto standard in the IaaS market and other providers try to be compatible to it, which reduces the risk of having too diverse solutions.

In a recent paper by Babar and Chauhan [3], they discuss experiences and observations from migrating an open source product to cloud. They have identified new quality attributes important for such migration; i.e. modularity and scalability as important for resource elasticity, portability of solutions, and a consistent view of system for external clients. The architecture of the open source solution (Hackystat [9]) had two major modifications which consisted of separating data from logic and adding an additional layer for orchestration of requests.

The state of the art analysis thus returned several interesting results that make the foundation of our work. However, we have to extend these with our model-driven approach taking advantage of OMG's ADM (Architecture-Driven Modernization [12]), KDM (Knowledge Discovery Metamodel [13]), SoaML (Service-oriented architecture Modeling Language [19]) and UML profiles such as in [11] in the recovery and service modeling phases. Project partners in REMICS are working on extending KDM and SoaML to cover concepts related to the migration to SOA and cloud computing paradigms. Activities relevant for our research project will therefore be added to the migration process. Additionally, the cloud paradigm introduces new software engineering challenges that are discussed later in this paper and may require new activities in the migration process that are not covered by the state of the art.

III. The REMICS Approach to Migration

The overview of the REMICS approach to migration is depicted in Fig.1. The baseline concept is the Architecture Driven Modernization (ADM) [12] by OMG. In this concept, modernization starts with the extraction of the architecture of the legacy application (the "Recover" activity). Having the architectural model helps to analyze the legacy system, identify the best ways for modernization and benefit from Model-Driven Engineering (MDE) technologies for generation of the new system. This information will be then translated into models covering different aspects of the architecture: Business Process, Business Rules, Components, Implementation, and Test specifications.

The above 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.

The migration process is supported by two complementary activities: "Model-Driven Interoperability (MDI)" in order to manage interoperability between services and "Validate, Control and Supervise" in order to guarantee that the migrated system provides the required functionality with the required Quality of Service (QoS).

While there are several methodologies for developing service-oriented systems from scratch or with focus on migration as discussed in Section II, REMICS has identified several gaps in the current state of the art that will be addressed in the project:

- Knowledge discovery is often limited to reverse engineering of legacy code. The business process and rules recovery is poorly addressed which are necessary for identifying services and designing new business processes.
- The architecture migration methods are mostly ad-hoc and lack a comprehensive methodology addressing dedicated design patterns and transformations. Especially migration to cloud is a new research area.
- Migration tools and methods need integration with model-based development methods.
- With many different platforms in the cloud and diverging technologies, we foresee the need for platform independent modeling combining SOA and cloud computing.
- There are no dedicated testing technologies for service clouds migration validation.

Addressing these gaps will be by developing methods, languages, transformations and tools. The REMICS methodology will integrate all these in an agile, model-driven service-oriented methodology for modernizing legacy systems.

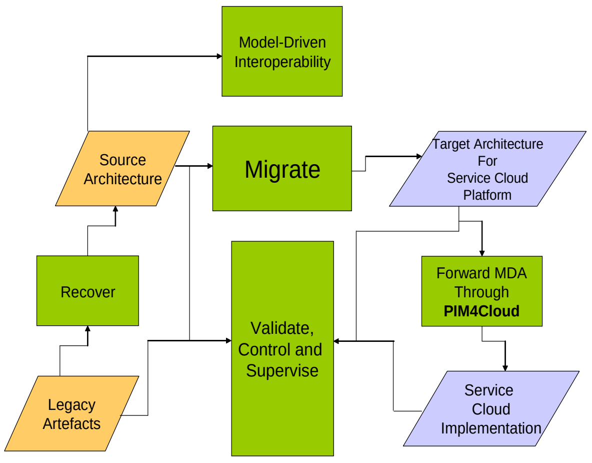
Model-Driven Interoperability
Source Architecture
Migrate
Target Architecture For Service Cloud Platform
Recover
Validate, Control and Supervise
Forward MDA Through **PIM4Cloud**
Legacy Artefacts
Service Cloud Implementation

Figure 1. The REMICS approach to migration

The methodology will define steps and guidelines for migrating these systems to loosely coupled systems deployed in a Service Cloud platform.

IV. Software Engineering Challenges

The first phase of the project has focused on analyzing the state of the art and understanding the legacy systems and their requirements. In this section we present how the Service Cloud paradigm introduces new challenges and impacts the activities that are necessary in a migration project. In some cases these are additional steps that may be added to activities already defined in existing software development methodologies. In other cases, new activities should be added that meet the challenges of the paradigm.

A. *Establishing the Context*

The first step in migration is to decide whether it makes sense to migrate a legacy system into services. The first step of the SMART process is to "establish the context" which includes understanding the business and technical context for the migration, identifying stakeholders, understanding the legacy system and identifying some candidate services for migration [18]. SMART provides a set of questions to answer which are relevant for our pilot cases as well and have been addressed in their descriptions of legacy systems. After this step one should determine whether the legacy system is a good candidate for migration.

We have performed an analysis of legacy systems and have identified some additional aspects that are not covered by SMART:

First, we take advantage of models and MDE methods. Therefore we focus on developing various models to describe the context regarding software architecture, business processes and deployment of systems for both the legacy system and the target migrated system.

Second, we felt that there should be more focus on the expectations of the stakeholders before taking a decision on whether to modernize the legacy system or not and what migration strategy to choose. We therefore added the following activities to the phase of establishing the context:

- *Describe disadvantages of legacy solution*. The problems should be identified in order to motivate migration. Examples from our pilot cases are maintenance expenses, scalability problems and the obsolescence of technologies.
- *Describe the expected benefits from migration*. These benefits should address the problems described above and how they will be solved in the Service Cloud paradigm, and may include additional benefits as well. Examples are a single point for installation and maintenance, improved performance and improved scalability.
- *Describe the disadvantages of the migrated solution.* The migrated solution may have disadvantages as well, for example no control of the infrastructure in a PaaS solution, unknown technologies, and less portability of the solutions.

The above information is important to gather before the feasibility decision point.

B. Modernizing the Software Architecture

Although the migration process in REMICS is in its initial phase, experimentation with cloud technologies and the state of the art analysis have highlighted some challenges that should be addressed regarding modernizing the software architecture of legacy systems.

The SOA paradigm implies the breakdown of architecture into high-grain components providing business services. These components should be loosely coupled and have clear functionality. It should be possible to reconfigure services or compose them in new ways to support new tasks. Modernizing the architecture of a legacy system to SOA thus requires developing models of new business processes, identifying services both top-down and bottom-up, and then deciding how to transform the current architecture to the migrated one, probably in a semi-automated way. The MOMOCS project has advised taking advantage of model-to-model transformations in this process [10]. We will also develop transformations when possible, as also done in the SINTEF project SiSaS (Scientific Software as a Service) and explained in [11]. However, the process requires intensive involvement of experts both in the recovery phase and in the migration phase and manual work as well. The SMART report has identified some challenges of migration such as separating business logic from APIs and changing the synchronous behavior of legacy systems to the asynchronous behavior of services which requires taking advantage of SOA patterns.

Adding the cloud paradigm introduces new challenges to the architecture modernization phase. Componentization of architecture should enable scalability and thus make possible the multiplication of a number of instances of the same component if taking advantage of IaaS technologies. On the other hand, when creating a cloud application, it is necessary to ensure that the application can run on a set of low performance resources, to cope with network latencies in a loosely coupled environment. The decomposition and decoupling aspect is thus similar to SOA while the scalability concerns are added. Babar and Chauhan mention that there is also need for new architecture evaluation methods since the existing methods do not emphasize cloud quality attributes such as scalability and accessibility [3].

Finally, the constraints introduced by cloud computing technologies should be considered. For the pilot cases in REMICS, we have evaluated both PaaS and IaaS solutions. The problem with the PaaS solution is that there is a significant dependence between the services these platforms offer and the client application. Thus these technologies may not be suitable for the migration of some legacy applications.

C. Modernizing Data

The legacy applications in REMICS are quite data-centric and data security is an important aspect. For taking advantage of data storage facilities provided by cloud computing technologies, the legacy systems databases should be modernized. One of the applications for example doesn’t have the three-layered model of Model-View-Controller and the business rules are in the database. Thus there is a need for separating business rules from data. In this process they will also remove redundant data and simplify the databases.

However, one of the concerns with introducing the cloud computing paradigm is data security. Therefore the companies should address the question of whether to take advantage of a private or public cloud and which data should be stored where. In one case, the databases will be deployed in a private cloud or be kept outside a public cloud solution. The second case targets a public cloud solution and depends on their security management.

D. Managing Non-Functional and QoS Requirements in the Cloud

The analysis of cloud computing platforms showed that these differ in characteristics such as load balancing, interoperability and convergence of the platform with other cloud computing platforms on the market, data storage system used by the platform, and fault tolerance mechanisms. Understanding the consequences requires more experimentation since there is yet little knowledge on these platforms. Therefore while selecting a cloud solution solves some problems of application providers, understanding these technologies is a new challenge. Requirements regarding servers, data storage and security, networking and response time should therefore be projected.

E. Verification and Validation in the Cloud

Each type of cloud (SaaS, PaaS and IaaS) abstracts the underlying layers and users do not have the possibility to control those. Also there are differences between public and private clouds regarding the degree of control. Babar and Chauhan emphasize the heavy reliance on cloud infrastructure providers and the fact that testing a software system deployed in cloud is different from testing traditional systems in many aspects [3]. Performance of the application depends on how effective virtual resources are managed by a cloud provider and testing should be performed on regular basis.

The major benefit of the public cloud from the application designer stand point is the ease of deployment, scalability and elasticity of the resources. However, the cloud comes with a cost; i.e. the application has to be capable to run on a set of relatively low performance servers. In a public PaaS solution, the scalability, elasticity and resources are managed by the PaaS provider and are out of the control of users. Thus the user needs to test the solution with these considerations and also monitor the cloud performance. At the IaaS level, scalability is managed by the application designer. However, the cloud paradigm allows new approach to scalability which is resource elasticity, i.e. the possibility to easily pop-up new instances of computation resources.

We therefore foresee to develop new methods for predicting the performance of applications and QoS in the cloud which will be addressed by the “Validate, Control and Supervise” work package in the REMICS project.

F. *Introducing Agility into the Migration process*

Our goal in REMICS is to provide an agile model-driven software engineering methodology to support migration to the Service Cloud paradigm. Agile methodologies are light, iterative, and with emphasis on continuous testing. They are popular in the industry and there exists several variants of them, although most for forward engineering of applications. However there is a dilemma between the large effort required in modernizing the software architecture and data early in the migration process and the agile approach of delivering software early. In the migration process, an iteration on architecture modernization is necessary before services are modernized part by part. We will address this challenge in the REMICS methodology and provide guidelines that combine the migration requirements with agile and iteration-based development.

G. *New Business Models*

Cloud computing technology providers have different invoicing strategies. While in principle the users pay for the resources consumed, this may for example be based on minutes or hours of usage or the number of transactions. Thus before selecting a provider, information regarding business model of the provider and the characteristics of the application using the services of a provider should be collected.

One of the motivations of companies for moving to cloud as service providers is to take advantage of the pay-per-use model. They should therefore develop new business models according to their context and the cost of services they use.

V. PILOT CASE IN REMICS

In this section we provide examples of the above challenges in the context of one of the pilot cases in REMICS.

DI Systemer (DISYS) is a Norwegian software vendor within the ERP (Enterprise Resource Planning)/Accounting/CRM (Customer Relationship Management) domain participating as a Small and Medium Enterprise (SME) partner in the REMICS project. DISYS has about 50 employees and a turnover of EUR 7 Mill in 2010. Key software products and services are CRM, accounting, payroll, invoicing, web portals and Application Service Provider. Fig. 2 shows the DISYS use case in REMICS.

The product portfolio is developed with different tools and languages and has evolved during decades. The following tools/languages are being used in the software development: COBOL, Delphi, C# .NET, ASP .NET, and UML for modeling. The software is consumed by the DISYS customers in different runtime environments: desktop standalone installations, and traditional Client/Server solutions in a Local Area Network (LAN). Some users also host the DISYS software in virtual machines executed in DISYS ASP centre or in their own data centre. There are practically no shared resources, i.e. there is one software installation per DISYS customer. Each customer has several users, typically in the range from 10 to 50. Accordingly, the present software portfolio comprises a conglomerate of software components of different kinds and exhibits a fairly high degree of interdependencies and few services, and a substantial part of the components are built upon legacy tools and technology.

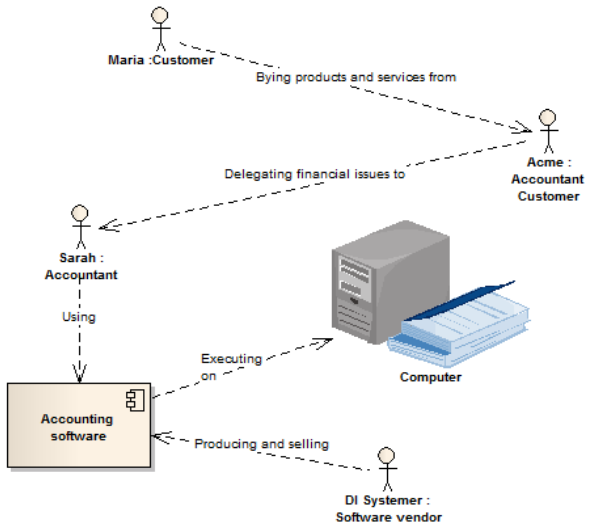
Maria :Customer
Bying products and services from
Acme : Accountant Customer
Delegating financial issues to
Sarah : Accountant
Using
Executing on
Computer
Accounting software
Producing and selling
DI Systemer : Software vendor

Figure 2. A simplified view of the DISYS use case in REMICS

The need for modernization of legacy software components has become more and more prevailing in view of partly poor legacy source code quality, problems with adding functionality or expanding software solutions, poor or missing documentation, and developers no longer working at the company.

The software modernization process at DISYS has been a bottom-up approach by re-implementing parts of the legacy COBOL programs step by step with more modern tools and languages. This modernization process started about ten years ago, where UML and Model-Driven Development (MDD) was introduced to generate Delphi source code as replacement for the legacy COBOL source code. However, there is still a considerable volume of legacy COBOL source code to modernize. The REMICS approach to software modernization is interesting in this context of several reasons. Firstly, the knowledge discovery process to represent complex and voluminous legacy COBOL code and enable inspection and manipulation in UML models is considered to be valuable because we would then be able to extract semantics and data structures hidden in the source code and maybe in the head of former employees. Secondly, our MDD experience gives us an expectation of the potential from the use of UML in the migration process. Thirdly, the deployment of a part of our legacy COBOL source to the service cloud is interesting, but also challenging, as discussed below.

In REMICS, we have performed an analysis of our case according to the process described in Section IV, under establishing the context. We have modelled the "as-is" and "to-be" business processes, software architecture and deployment. The main disadvantages of the current solution are identified to be:

- The report consumers do not have direct access to the system to initiate reports or change them according to their needs;
- DISYS software developers must develop and maintain reporting software in different software projects and programming tools;
- New and amended reports and report layouts must be programmed and installed at the user's runtime environment as a part of a general software update;
- The sales department must keep track of several software products;
- There is no simple way to integrate reporting with data from 3rd party data providers.

We expect the following advantages of the target solution as SaaS and deployed in cloud:

- Obtain one reporting program solution common for all our deployment platforms;
- Distribute new reports to users without having to compile and deploy new programs to every single customer installation, i.e. deploy reports to one single location, e.g. Windows Azure;
- Enable layout customization to customers without involving development support from software vendor;
- Web based reporting: there should be no need to install anything else than a web browser to retrieve reports;
- Enable report consumption on mobile devices;
- Include new users of DISYS software and increase knowledge of company brand (market value);
- Replace the numerous reporting program installations (one per customer) with one deployment in the service cloud;
- Save costs by reducing the need for support during installation and use;
- Meet scalability issues by requesting more hardware from the service provider when needed;
- Enable payment per use.

An additional motivation for participating in the REMICS project is to experiment with new technologies in order to understand their impact on our market.

We don't see any disadvantages in modernizing our software architecture to SOA while the cloud technologies may offer some challenges, especially regarding portability if we are not satisfied with one platform.

Specific software engineering challenges in our case are so far identified to be:

A. *Modernizing the Software Architecture*

Our software components are relatively highly integrated and exhibit a high degree of interdependency. For that reason, we have identified legacy software components that could be modernized and migrated to the service cloud without dependencies from the service cloud to any other remaining software components in the traditional runtime environment. In order to obtain that, we have to replicate reporting data from the existing installations at the bookkeeper's sites where the transactions actually are generated up to the service cloud site.

We have defined the migrated software architecture as a combination of a modernized database in Microsoft SQL, Web-services and Asp .NET components.

B. *Selecting a Suitable type of Cloud based on Data Management and Transfer Requirements*

The data replication issue is a concern, since we have to deal with quite large amounts of data uploaded from the accountant's site via the Internet up to the data storage in the service cloud. Just for one single customer, the account history could constitute several millions rows of financial transactions. A deployment of the migrated system in a public cloud could therefore be a challenge with regard to bandwidth during data upload and possibly scalability during data storage. However, the scalability and bandwidth issues from data querying and report production are not considered to constitute a problem. The data is aggregated during the query operations and report generation and represent a low data volume do be returned to the user's browser compared to the data uploading process.

As a consequence of the bandwidth concern, we considered a private cloud deployment as an alternative. DISYS is serving its own data centre for the customers and data replication would be local within the data centre. However, also this deployment solution could become a challenge if the reporting centre was offered to other DISYS software users not hosted in our data centre. In such a case, their data would have to be replicated from their respective account data tables and up to the private service cloud via the Internet. Hence, we may face a similar bandwidth problem with the private cloud as with the public cloud. Database scalability could also become a problem from high loads during data storage in particular because of database server license costs, additional hardware etc.

A private cloud deployment will to a certain degree simplify security against data theft since DISYS can move and manipulate data within a more controllable environment than a public service cloud. However, a private cloud deployment still has to maintain data security for reporting data between respective customers, i.e. accounting data from one customer must not be available to another. In a public service cloud deployment however, DISYS rely totally on the security offered at the public cloud platform.

C. Managing Interoperability Issues

One of the goals of the migration to the Service Cloud paradigm is to facilitate integration of our data with 3rd part data providers. This is in line with the vision of SOA and cloud computing paradigms to allow new service chain models. There is accordingly a need for defining a data mapping from numerous source data providers to a model representation at the report data storage in the migrated system, possibly via a standardized canonical model. Contrary to the COBOL data structures in the legacy data storage of the DISYS programs, the third party data provider data storage is a MS SQL database. This challenge is to be addressed by the research in REMICS on model-driven interoperability.

D. Specifying QoS Requirements

It is difficult to quantify the total number of simultaneous requests to the migrated system, but as a coarse estimate we could say that the number of simultaneous requests from data upload by the accountants could constitute 1000 instances whereas the number of simultaneous requests from report consumers could be estimated to 200. Both estimates are given for a single report server installation serving all customers of DISYS. When the migrated system has been established and put into use, the volume of data stored would increase steadily whereas the number of requests is expected to remain stable both from the data providers and the report consumers.

Loss of data in the reporting SQL databases is not critical, since the database may be reinstated by replicating data again from the data provider (the accountants COBOL data storage).

E. Testing the Migrated Solution

We should test the migrated solution to verify that the functionality and data integrity has been preserved during the knowledge discovery and migration process. One way to perform this is to compare the content of the data structures which are actually used in generation of the report per se.

Testing the performance, load-balancing and security of the migrated solution is a concern since the results depend on the Internet bandwidth, load-balancing and security mechanisms of the cloud service provider. We have to identify scenarios for testing and understand the technologies better during the implementation and validation phases.

VI. CONCLUSIONS AND FUTURE WORK

In this paper we have presented some software engineering challenges that are identified so far in the REMICS project related to the migration of legacy applications to the Service Cloud paradigm. One problem is that the cloud computing technologies are proprietary environments that will require great effort to understand the technologies involved and constraints placed on service consumers and providers. IT organizations roles in the value chain change since they will be more and more service consumers that depend on the availability and performance of the services provided by cloud technology providers.

Some identified software engineering challenges are related to establishing the context and understanding the technologies and business models, while others are related to the modernization step of legacy applications (software architecture as well as data) and testing the solutions. New quality requirements such as scalability and storage become important in the migration and the service users should be able to project their requirements regarding these. The goal of REMICS is to provide an agile, model-driven, tool-supported methodology that takes advantage of the state of the art and includes additional steps when necessary.

Research in REMICS continues in the coming months with the extraction of models from legacy code and modernizing the architecture of the two pilot cases. In parallel we work on the methodology that will be accessible on the project website and also implemented in the Eclipse Process Framework (EPF).

ACKNOWLEDGMENT

This work has been partly funded by the European Commission through the REMICS project, contract number 257793, within the 7th Framework Program. We acknowledge the contributions of consortium members participating in the state of the art analysis, especially Andrey Sadovykh and Antonin Abhervé from SOFTEAM, Franck Barbier from Netfective Technology and University of Pau, and Gorka Benguria from Tecnalia. This work is also funded by the SiSaS project at SINTEF. SiSaS focuses on the migration of scientific software to services.

References

[1] Amazon EC2; http://aws.amazon.com/fr/ec2/

[2] A. Arsanjani, S. Ghosh, A. Allam, T. Abdollah, S. Ganapathy,and K. Holley, "SOMA: a method for developing service-oriented solutions", IBY Systems Journal, vol. 47(3), 2008.

[3] M.A. Babar, M.A. Chauhan, "A tale of migration to clou computing for sharing experiences and observations", to be published in proceedings of Software Engineering for Cloud Computing Workshop at ICSE 2011, 2011.

[4] J. Butler, "The architecture component of the SAE reference framework for SOA", CBDi Journal, http://www.cbdiforum.com/secure/interact/2007-03/the_architecture_component.php, March 2007.

[5] S. Comella-Dorda, K. Wallnau, R.C. Seacord, J. Robert, "A survey of legacy system modernization approaches, Technical Note No. CMU/SEI-2000-TN-003): Carnegie Mellon Software Engineering Institute, 2000.

[6] Eucalyptus; http://open.eucalyptus.com/

[7] Gartner reserach, URL http://www.gartner.com/research/spotlight/asset_112573_895.jsp, last visited in June 2004.

[8] Google App Engine; http://code.google.com/intl/en/appengine/

[9] P.M. Johnson, "Requirements and design trade-offs in Hackystat: an in-process software engineering measurement and analysis system", 1st International Symposium on Empirical Software Engineering and Measurement, IEEE Computer Society, pp. 81-90, 2007.

[10] MOMOCS project website; http://www.momocs.org/

[11] J. Oldevik, G.K. Olsen, U. Brönner, N.R. Bodsberg, "Model-driven migration of scientific legacy systems to service-oriented architectures", 1st International Workshop on Model-Driven Software Migration, 4 pages, 2011. URL http://userpages.uni-koblenz.de/~ist/mdsm/2011/index.php

[12] OMG ADM; http://adm.omg.org/

[13] OMG KDM; http://www.omg.org/technology/documents/modernization_spec_catalog.htm

[14] OpenNebula; http://opennebula.org/

[15] OpenStack; http://www.openstack.org/

[16] M. Razavian, P. Lago, *"Understanding SOA migration using a conceptual framework"*, Czech Society of Systems Integration, 2010.

[17] REMICS project website; http://remics.eu.

[18] SMART report by SEI. URL http://www.sei.cmu.edu/library/abstracts/reports/08tn008.cfm, last visited in March 2011.

[19] SoaML; http://www.omg.org/spec/SoaML/

[20] Windows Azure; http://www.microsoft.com/windowsazure/

[21] O. Zimmermann, P. Krogdahl, and C. Gee, "Elements of Service-Oriented Analysis and Design, an interdisciplinary modeling approach for SOA projects", URL http://www-128.ibm.com/developerworks/webservices/library/ws-soad1/, last visited in June 2004.