Towards SaaS adoption: What are the options from a Legacy System point of view?

Legacy Systems are deemed a pariah in the industry, from a maintenance perspective, but more often than not they hold a majority of the companies assets and can't simply be tossed aside, which forces all kinds of companies to face the following dilemma: should we attempt to modernize an aging software infrastructure or re-write it from scratch? The former is the most sensible option, since what characterize a legacy system, is that the initial programmers and/or architects are no longer part of the organization, and left with a majority of the knowledge of the design decisions made and the some valuable expertise about the system. It is a well-known fact that this question is hard, and when choosing any of the answers, the solution will be a very challenging task to accomplish. Therefore, there is a need to present tools and methodologies that can accompany these companies through this process. Optimally, a single solution would guide and assist from end-to-end with the process of modernization of a Legacy System. But reality is far more complex than this, and such concept of a solution is far from reach.

Once a decision is made to modernize a Legacy System, further considerations must be taken, such as: What is the new targeted architecture? Will it run on the same platform as before? But more specifically we want study what is available to take a Legacy System to the Cloud infrastructure as Software as a Service. SaaS is defined as the delivery model, and the Cloud as the infrastructure hosting and providing these delivery capabilities. Therefore, we will take a look at 3 of the main frameworks that have been proposed in the last decade to perform this kind of task and how they fare against each other.

There is quite a division between all of these frameworks or methodologies, one in terms of Migration and Modernization. Migration frameworks consist of trying to provide the tools and techniques to ease the migration from operating environment to another that is deemed more powerful and extendable. Conversely, Modernization is concerned with transforming the legacy system to a modern computer programming language using up-to-data and state of the art libraries, protocols and/or hardware components. More importantly, it aims at extending but also retaining the value of the original legacy system and diminishing maintenance by adopting newer software engineering techniques and methodologies [1]. Importantly, the distinction between those two is not exclusive, in the sense that some frameworks may contain a modernization phase followed by a migration phase.

The 3 frameworks we are going to look at are: REMICS, ARTIST and MILAS. All of them spun from growing interest to provide means to different companies that would help them embrace the new off-site paradigm shift that is Cloud Computing. We will present them sequentially and then proceed to a comparison.

REMICS

The acronym, REMICS, stands for **R**euse and **M**igration of legacy applications to **I**nteroperable **C**loud **S**ervices. The research project, which is supported by the European Commission, started in 2010 and lasted 36 months. Its primary objective is to develop a set of model-driven methods and tools that supports organizations with legacy systems to modernize them according to Cloud Paradigm in conjunction with a SOA aiming at the SaaS model [2].

They use Architecture Driven Modernization (ADM) by OMG, as a baseline for their approach. ADM is a process of understanding and evolving existing software assets for the purpose of modernization as we defined earlier [3], focusing on the system's architecture and the ability to transform it to the target architecture (in a standardized way) [4]. The figure below shows the REMICS approach to migration.

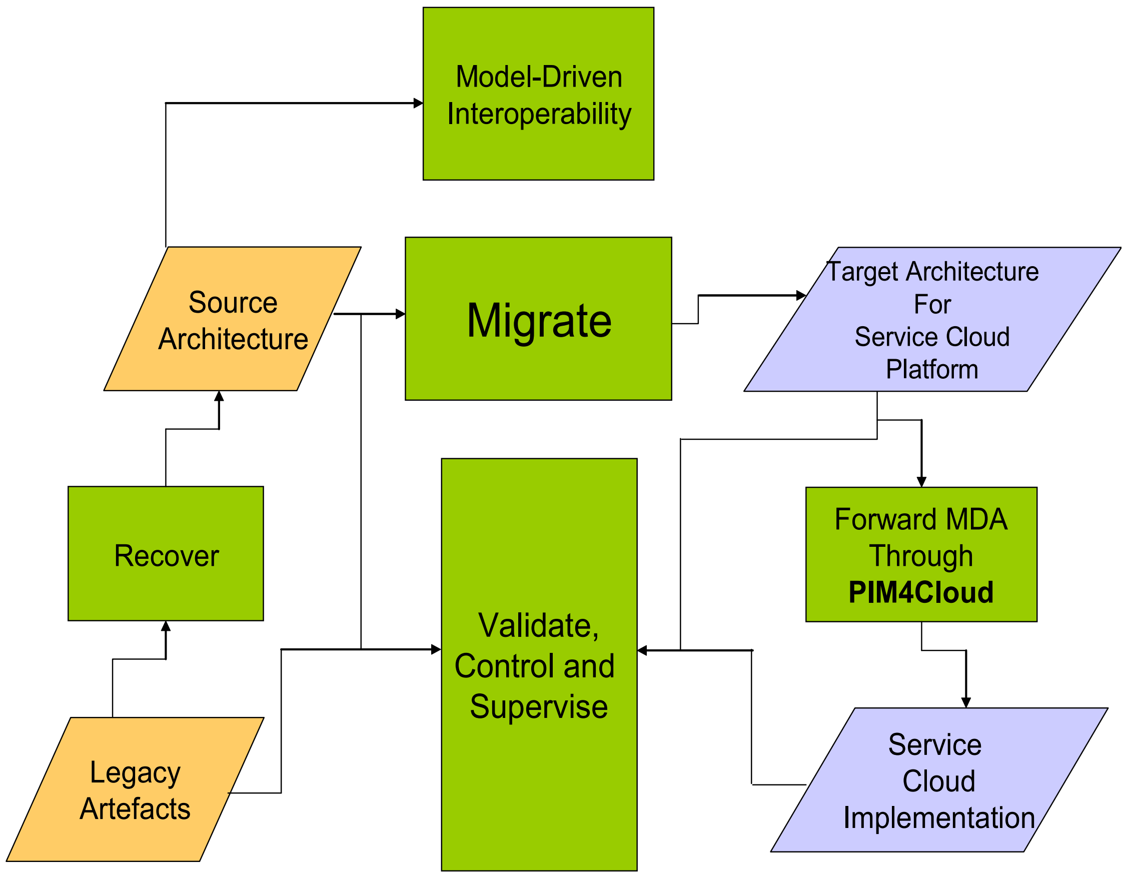
Model-Driven Interoperability
Source Architecture
Migrate
Target Architecture For Service Cloud Platform
Recover
Validate, Control and Supervise
Forward MDA Through **PIM4Cloud**
Legacy Artefacts
Service Cloud Implementation

Figure 1: REMICS: Migration approach

We can observe to major activities to this approach, namely Recover and Migrate [5].

Starting the modernization effort by extracting the current architecture of the legacy application from the different Legacy Artefacts. This helps to gain more insight as to which modernization techniques will be best suited for this particular system and how can it benefit from a the different Model-Driven Engineering (MDE) technologies currently available. This activity is performed using Knowledge

Discovery techniques, which consist, almost exclusively, to reverse-engineer the existing legacy code. Once completed, the knowledge will be represented using REMICS Knowledge Discovery Metal-model (KDM), which encompass the Business Processes and Rules. The components will be expressed using SoaML, which is a UML profile and meta-model for the modeling and designing of services within a Service Oriented Architecture (SOA). Finally, the current implementation of the system will be represented using standard UML. All of these models together, as a whole, will serve as the Source Architecture.

Following the Recover activity is the Migrate activity, which makes use of the models created in the previous activity as a starting point. Then the new architecture for the system will be designed applying specific SOA and cloud computing patterns, and by performing Legacy Components Wrapping. Also by performing Legacy Components replacement and adding service discovery and finally by opting for a design based on the service composition.

We can also observe to complementary activities to the Migrate activity, which are Model-Driven Interoperability and Validate, Control and Supervise. These two complementary activities are essential to guarantee that the migrated version of the legacy system provides the required functionalities and that it does it with respect to the required Quality of Service (QoS). More specifically, Model-Driven Interoperability provides insights in terms of services discovery and adaptation.

This results in target architecture for service cloud platform, which consists of SoaML, with REMICS extensions for Service Clouds, model of the migrated system that also provides traceability to the Business Models.

The final activity consists of using the MDA that was just created as an input to PIM4Cloud, which [6] defines it as: *"… a meta-model and UML profile for describing IT systems deployment to cloud platforms from an application designer's perspective."* Thus it will perform model transformation and will lead to code generation, which will provide full traceability of the migrated system. Finally, the migrated system can be deployed on any cloud platform.

ARTIST

Advanced Software-based Service Provisioning and Migration of Legacy Software (ARTIST) project aims at providing tools and techniques to reverse engineer legacy software into models from which software that is compliant with the cloud infrastructure requirements can be generated. The authors in [7] describe some of the major outcomes of the ARTIST project as being the following: it covers a migration method and a comprehensive tool suite to support modernizing legacy software towards cloud-based technologies. But also, ARTIST will provide a model-based reverse and forward engineering tool-chain created around the concept of an open development environment. One of the key features of ARTIST is the concept of a central repository to store all the Modernization artifacts. This will permit reusability of these artifacts across all the different modernization tasks and will create a knowledge source from which all the scenarios and best practices, in terms of software modernization, can be derived.

The modernization process that they propose is composed of three distinct phases: pre-migration, migration and post-migration.

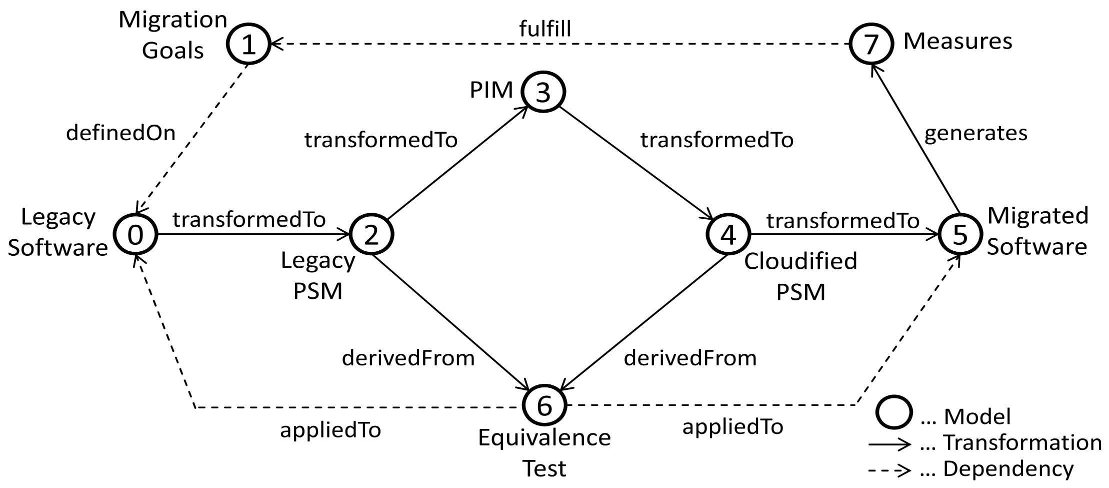

Figure 2 ARTIST: Modernization Process

ARTIST modernization process is model-based, and all the different steps are numbered and represented in the figure below.

The pre-migration phase consist of an analysis of the Legacy Software in both technical and non-technical perspective concerning the different migration strategies. As a result of this analysis are the Migration Goals, which will dictates how the migration should be performed in the subsequent phases. Next, they reverse-engineer Platform Specific Models (PSMs) from the legacy software. These models contain all the specifics about the platform that the current implementation of the legacy software is deployed on. From these models, the derived more abstract models named Platform Independent Models (PIMs), from which they try to remove any dependencies of the old platform and thus isolating the legacy system from it's implementation.

Then the migration phase starts. They use the migration goals to choose the appropriate transformations to apply to the PIMs, which consists of applying optimization patterns and integrating the underlying cloud-specific modernization opportunities contained in the initial models. This step results by generating a model-based representation of the migrated software. This representation encompasses the platform-specifics that correspond to the selected cloud environments and thus it is called "Cloudified PSM". Subsequently, using the Cloudified PSM, they generate an executable of the migrated software that is going to be hosted in a cloud environment, which concludes the migration phase.

Finally, the post-migration phase they derives Equivalence Tests from the model-based representations of both, legacy and migrated software which that the migrated software is functionally equivalent to the original legacy software. The final step is to assess that the migration goals are fulfilled, and is accomplished by analyzing the execution of the migrated software, to evaluate non-functional properties, and obtain the measures in order to perform the comparison to the migration goals.

MILAS

Hans Aage Huru presented, perhaps one of the older frameworks, as a master's thesis back in 2009 [8]. MILAS stands for **M**odern**I**zing **L**egacy **A**pplications towards Service Oriented Architecture (SOA) and Software as a Service (**S**aaS). The author presents it as a generic methodology, which helps developers to migrate a legacy system to a modernized system following the SOA/SaaS paradigm. This methodology also uses a model-based approach to modernization of the legacy system to be migrated. It uses the horseshoe model [9], defined by SEI, as a baseline to define its methodology.

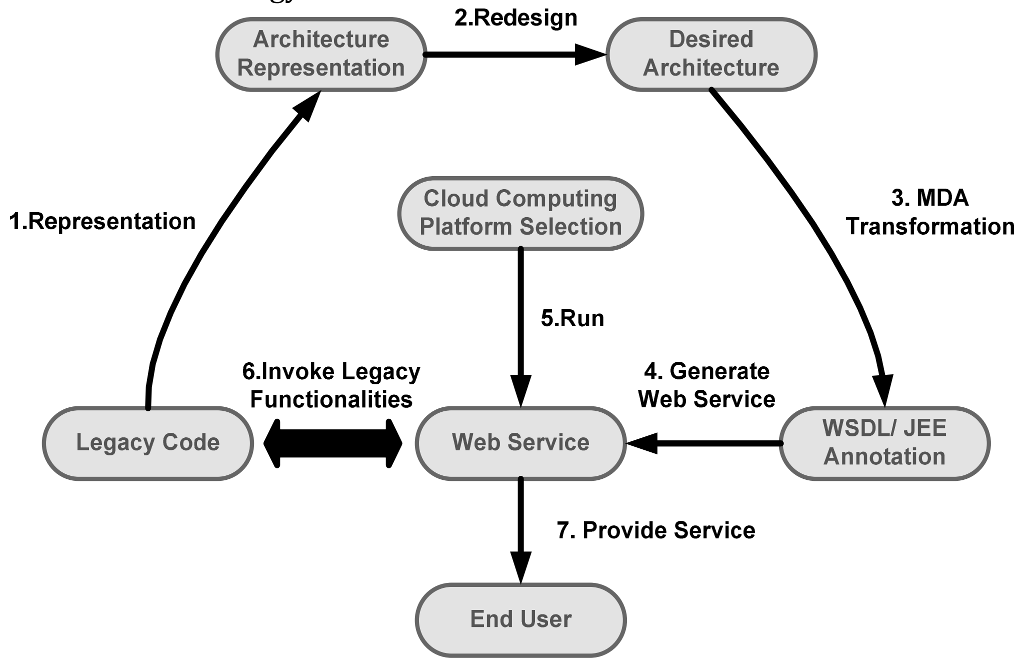

Figure 3 MILAS: Methodology Overview

The figure above provides an overview of the methodology proposed by MILAS, which contains 7 distinct steps. The first step consists of creating an architectural representation of the legacy application based on the analysis of the source code and the diverse textual documentation available for this legacy system. The second steps consist of redesigning the original architecture model and identifying the services that can be provided in a SaaS architecture, and derive a SoaML model that will represent the desired architecture. The third step consists of applying MDA available technologies to transform the architecture from SoaML (or SysML or UML) to target code like WSDL and/or JEE Annotations. The fourth step consists of generating the web services described by the WSDL and/or JEE Annotations generated in the previous step. The fifth step is concerned with the invocation of legacy functionalities from a web service-base perspective. More specifically, the service-base application invokes the functionalities from the identified functions and service points in the legacy application. The following figure provide a graphical representation of this step, for brevity purposes we won't describe it but rather simply present it.

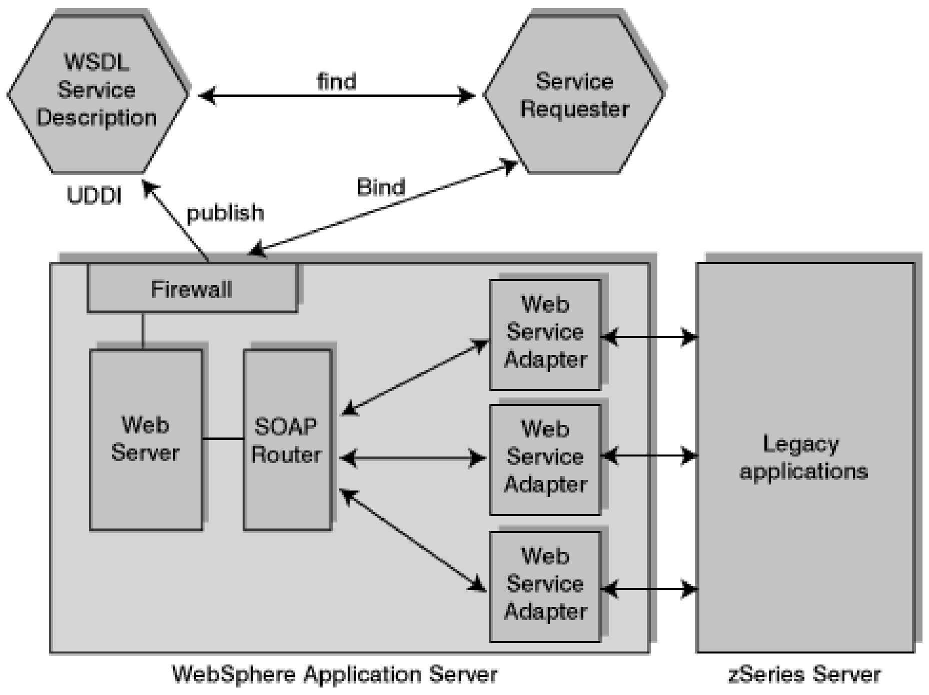

Figure 4 MILAS: Legacy Applications to Web Service

The sixth step simply consist of choosing the most suitable cloud computing platform (SaaS, PaaS or IaaS) according to the specific requirements of the target system and to support the execution of the web services defined earlier. The final step consists of the actual deployment to the cloud and enables the end users to consume the legacy functionalities through web services.

This methodology provides much depth in the different steps stated above, but for brevity purposes we simply presented them.

Comparative evaluation

In this section we will proceed to a comparative evaluation of the 3 frameworks presented and enumerate the differences and the various features. We will use the requirements described in [8] as a metric to compare how each one fare against each other. Therefore, let's start by presenting, said requirements.

Huru presents the criterions for a modernization approach considering SOA and SaaS as the following 4 steps: Recovering, Migration, Service Modeling (PIM-PSM) and Platform (PSM-Realization). We will present them in a table and proceed to explain them with the results for the different frameworks.

Table 1 Evaluation of the different Methodologies/Frameworks

	REMICS	ARTIST	MILAS
Recovering			
Recover to architectural representation	Yes	Almost	Yes
Analysis of Legacy Architecture	Yes	Partially	Yes
Retrieving Requirements	Yes	No	Yes
Migration			
Refactoring for SOA	Yes	No	Yes
Refactoring for SaaS	Indirectly	Yes	No
Control your work (Analysis)	Yes	Partially	Yes
Traceability between models	Yes	Yes	Yes
Service Modeling (PIM-PSM)			
Modeling for SOA and MDA	Partially	Implicitly	Yes
Modeling for SaaS	Yes	Yes	Partially
Modeling service mediation	Yes	Yes	No
Platform (PSM-Realization)			
Transform to SOA	Yes	No	Partially
Transform to SaaS	Yes	Yes	Partially

Recovering

Let's start with the first criterion for the Recovering requirement: Recover to architectural representation, which the author (Huru) describes as the ability to generate an Abstract Syntax Tree (AST) of the Legacy System's source code, and to analyze it in order to build an architectural view of the Legacy System. REMICS starts with the extraction of the architecture of the Legacy System, and therefore fulfill this criterion. ARTIST is rated as Almost for this criterion, since it does not perform Architectural extraction from the legacy source code per se, but rather uses reverse-engineering techniques to extract PSMs from the Legacy System, which to an extent reflects some architectural characteristics. MILAS proposes to extract information from the Legacy System and transform it into a Knowledge Discovery Meta-model (KDM) and AST, thus fulfilling this requirement for this criterion.

The second criterion is described as the ability to extract information out of the recovered architecture to further the understanding of the source code and reconstruct the architectural overview. REMICS proposes to use reverse-engineering techniques such as KDM to generate models that represents the current architecture and also proposes to analyze these models to assess the feasibility of this effort, thus it fulfills this criterion. ARTIST doesn't contain any distinct step for the architectural analysis of the recovered engineering artifacts, but rather proposes an initial analysis at the beginning of the process that supports the definition of the migration goals, thus it score Partially since indirectly some architectural analysis is done. MILAS fulfills this criterion since it uses an architecture representation that conforms to an open standard, numerous tools are available to perform the necessary architectural analysis.

The last criterion of this requirement consist, as it names implies, of retrieving the legacy business requirement and combining them with the new requirements from the business management. REMICS proposes, in its Recover phase, to retrieve the essence of the IT system by using methods that are based on KDM extensions [10], thus it fulfills this criterion. ARTIST being heavily model-driven doesn't explicitly recover, or attempts to, any requirements per se, but rather rely on the assumption that these will be contained within the PIMs derived from the PSMs extracted from the Legacy System and to perform verification of the requirements compares them against the models of the Migrated System, and therefore doesn't fulfill this criterion. MILAS proposes to use Business Process Model Notation (BPMN) and Business Motivation Models (BMM) to represent the requirements and then verification can be performed against the new architecture linking them throughout the different modeling levels, and thus fulfills the criterion.

Migration

In the Migration requirement, the first criterion consist of refactoring the code to a SOA system using good OO design patterns in order to meet the requirements in terms of reusability, maintainability and portability. Also, SOA requires that the architecture exhibit high-cohesion and low coupling between its compositional components. REMICS has for objective for its Migrate phase to re-architecture the Legacy System to a dynamically reconfigurable SOA. It propose to do so by using monolithic component replacement (when possible), or trying to find automatically generated mediators to adapt the existing services via Model-Driven Interoperability (MDI), or finally for the re-development component candidates wrap them into a SOA packaging. Thus it fully addresses this criterion. As for ARTIST, it doesn't address the idea of progressive refactoring, (from Legacy to SOA then to SaaS), but rather provide means to go directly to refactor directly for a cloud environment. It is understood that some of the SOA requirements and non-function requirements will be met, but it isn't explicitly clear as to why or how this is done. Therefore, it does not fulfill this criterion. MILAS proposes to use 13 steps to achieve a sound OO design that act as a foundation for the SOA design, and thus by induction it provides a SOA architecture for the Legacy System which fulfills this criterion.

The following criterion is very similar to its preceding counterpart, but is now concerned with SaaS, thus refactoring the code to a SaaS delivery model. REMICS addresses this criterion because what they proposed for the previous criterion indirectly leads to a SaaS refactored source code built on a SOA that is dynamically reconfigurable to supply more services, run more instances, at peaks and thus is cloud compliant. ARTIST propose to apply transformations the PIM, derived from the PSM of the Legacy System, into PSMs that are compatible with the cloud environments, which are referred as "Cloudified PSMs". These transformations consist of applying optimization patterns and integrating cloud-specific modernization opportunities. Thus it fulfills this criterion. MILAS does not address this criterion explicitly, but rather implies that by providing a strong SOA for the Legacy System makes for a basis to the SaaS delivery model but doesn't provide any means to do so, and rather left some SaaS requirements unaddressed.

This criterion consist of the ability to perform proper analysis of the generated models (UML and AST), and attest that the new system conforms in terms of re-usability and manageability, also in terms of robustness. REMICS proposes an complementary activity to the Migrate phase, which consist of validating the models created against the original to assess that the functionalities are preserved but also in terms of specifications, and proposes to use model-base testing for the functional aspects of the migrated system. For the validation and verification of the specifications they propose to use a set of rules, various verification and validation processes and observe if they meet specifically defined goals. Thus it fulfills this criterion since they propose a complementary activity that can be executed periodically to assess conformance. ARTIST doesn't seem to provide any means to periodically assess progress nor conformance, but since it uses MDE, periodical partial model testing can be done using different tools available, given that the models are expressed in a standardized format. Thus it partially fulfills this requirement. MILAS fulfills this criterion by relying on the fact that the target and the source architecture are expressed using UML and ASTM/KDM representation. This means that some tools are available to perform analysis, and assumes mass adoption of these standards from an industry perspective in order to provide state of the art tools to complement these analyses. Thus it fulfills this requirement.

The final criterion of this requirement consists of the ability to provide and/or achieve traceability between models. REMICS is heavily based on MDE; therefore non-functional and functional requirements that were present in the Legacy System can be traced through the different models generated by this process all the way to the migrated system, using the various model representations of the different components. This is very similar and aligned with the Validate complementary activity. Thus it fulfills this criterion. ARTIST fulfills this criterion in a similar fashion, by using standardize model representation for all of it steps outputs it can achieve traceability. And this is also true for the MILAS framework.

Service Modelling (PIM-PSM)

The first criterion of this requirement is concerned with ability of generating models for SOA and MDA. REMICS technically provides models that reflects the architectural design while exhibiting the peculiarities of the delivery models, SaaS in this case, and thus makes no distinction between the two and does not embrace the progressive modeling approach that is propos by Huru, thus partially fulfills this criterion. ARTIST does not address this criterion, but rather proposes a process that transforms the Legacy System extracted PSMs to PIMs and then from PIMs to cloudified PSMs that addresses directly cloud platforms requirements and therefore implicitly, since a SaaS application is conceptually SOA, it fulfills this criterion. MILAS fulfills this criterion since all the models used for the different representations are well suited for SOA modeling, such as UML, SoaML, BPMN and BMM.

The following criterion addresses a similar concern, but this time relating to SaaS, therefore the ability to generate models for SaaS. REMICS fulfills this criterion since it proposes to forward the PIMs through PIM4Cloud, which is a SoaML extension that takes a MDA, a SoaML model, and perform cloud specific transformation to output a cloud-compliant model from which code can be automatically generated. ARTIST proposes a similar technique to achieve their Cloudified PSMs from the PIMs, which then can be used to generate the code of the migrated software automatically. Thus it fulfills this criterion. MILACS on the other-hand proposes PIM4Wrapping, a novel UML profile that provide means for defining services from executables by wrapping them. By using this profile, some of the requirements in terms of SaaS can be met by wrapping legacy components to make them interact in a SaaS manner. Thus it partially fulfills this criterion.

The final criterion of this requirement is about modeling the service mediation. By this it is intended to provide means for different services to communicate and map the different format of messages used by the source service to the desired format for the target service, thus providing message agnostic communication between services. REMICS provides extended service mediation and data interoperability within the PIM4ServiceInteroperability, and for brevity purposes we will refer to [11] for more details, and thus fulfills this criterion. ARTIST provides means to perform service mediation through their CloudML@ARTIST [12], which is an extension of the CloudML. CloudML is a modeling language: *"... that facilitate the specification of provisioning, deployment, and adaptation concerns of multi-cloud systems at design-time and their enactment at run-time. [13]"* and thus it fulfills this criterion. MILAS doesn't provide any means to do service mediation but rather directs the user to use a UML or SoaML, if such exist, to achieve it. Thus by itself it doesn't fulfill the criterion.

Platform (PSM-Realization)

The first criterion of this requirement relates to the ability to transform the PSMs generated from the modernization/migration effort into platform specific code that can be deployed on a SOA platform such as JBoss SOA suite [14]. REMICS proposes a component that is integrated with the code generators provided by Modelio, which is an open-source modeling environment [15], and is called Dynamic Core. It uses the various models generated from the previous steps and it generates code that conforms to the SOA. ARTIST does not address this perspective. MILAS partially fulfill this criterion, since it proposes to use MOFScript to generate platform specific code that is derived from the models, but does not provide any means to do it and simply states that it can be done.

The final criterion is identical to the previous one with the exception that the code is aimed at a SaaS platform instead, such as Amazon EC2 or Google App Engine (GAE). REMICS uses the PIM4Cloud profile that they created to tailor their package application to the specificity of the cloud service platforms by using a Domain Specific Language (DSL), and thus fulfills this criterion. ARTIST proposes Cloud Application Modeling Language (CAML) that draws from the same rationale as PIM4Clouds but provides a Cloud Deployment Modeling Library (CDML) [17], which allows representing cloud-based deployment models independently from particular service providers, and thus fulfills this criterion. MILAS partially fulfills this criterion for the same reasons as the previous criterion.

References

[1] Gardner, D: "Not just a nip and tuck, application modernization extends the lifecycle of legacy code assets", [1] ZDNet, October 24, 2006

[2] Mohagheghi, P., & Sæther, T. (2011, July). Software engineering challenges for migration to the service cloud paradigm: Ongoing work in the REMICS project. In Services (SERVICES), 2011 IEEE World Congress on (pp. 507-514). IEEE.

[3] Newcomb, P. (2005, November). Architecture-driven modernization (ADM). In Reverse Engineering, Working Conference on (p. 237). IEEE Computer Society.

[4] Pérez-Castillo, R., de Guzmán, I. G. R., & Piattini, M. (2010). Architecture-Driven Modernization. Modern Software Engineering Concepts and Practices: Advanced Approaches, 75.

[5] The REMICS model-driven process for migrating legacy applications to the cloud, (ppt), Marcos Almeida, Research Engineer at SOFTEAM, PMDE-ECMFA 2013.

[6] Sadovykh, A. A. A., Srirama, S., Jakovits, P., Smialek, M., Nowakowski, W., & Chauvel, F. (2012). Deliverable D4. 1 REMICS PIM4Cloud.

[7] Bergmayr, A., Bruneliere, H., Izquierdo, J. L. C., Gorronogoitia, J., Kousiouris, G., Kyriazis, D., ... & Wimmer, M. (2013, March). Migrating Legacy Software to the Cloud with ARTIST. In Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on (pp. 465-468). IEEE.

[8] Huru, H. A. (2009). MILAS: ModernIzing Legtacy Applications towards Service Oriented Architecture (SOA) and Software as a Service (SaaS).

[9] Bergey, J., Smith, D., Weiderman, N., & Woods, S. (1999). Options analysis for reengineering (OAR): Issues and conceptual approach (No. CMU/SEI-99-TN-014). CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST.

[10] Mohagheghi, P., Berre, A. J., Sadovykh, A., Barbier, F., & Benguria, G. (2010). Reuse and migration of legacy systems to interoperable cloud services-the REMICS project. Proceedings of Mda4ServiceCloud, 10.

[11] Sadovykh, A. A. A., Srirama, S., Jakovits, P., Smialek, M., Nowakowski, W., & Chauvel, F. (2011). Deliverable D5.1 REMICS PIM4ServiceInteroperbility extension to SoaML.

[12] ARTIST FP7-317859, D7.2.1 – Cloud services modeling and performance analysis framework, Version 1.0, 2013.

[13] Alessandro Rossini, Arnor Solberg, Daniel Romero, Jörg Domaschka, Kostas Magoutis, Lutz Schubert, Nicolas Ferry, and Tom Kirkham. D2.1.1 — CloudML Guide and Assesment Report. PaaSage project deliverable, October 2013.

[14] RED HAT JBOSS FUSE SERVICE WORKS Datasheet, Red Hat, 2014.

[15] Modelio, http://www.modelio.org/

[16] Sadovykh, A. A. A., Srirama, S., Jakovits, P., Smialek, M., Nowakowski, W., & Chauvel, F. (2013). Deliverable D4.5 REMICS Migrate Principles and Methods.

[17] ARTIST FP7-317859, D9.2 – Modeling Language and Editor for Defining Target Specifications, Version 1.0, 2013.