Cloudifying Applications with ARTIST
A Global Modernization Approach to Move Applications onto the Cloud

Leire Orue-Echevarria[1], Juncal Alonso[1], Hugo Brunelière[2], Andreas Menychtas[3], Philip Langer[4] and Manuel Wimmer[4]
[1]*TECNALIA, ICT-European Software Institute Division, Parque Tecnológico Ed #202, E-48170 Zamudio, Spain*
[2]*AtlanMod Team Inria, Mines Nantes & LINA, Nantes, France*
[3]*National Technical University of Athens, Athens, Greece*
[4]*Vienna University of Technology, Vienna, Austria*
{leire.orue-echevarria, juncal.alonso}@tecnalia.com, hugo.bruneliere@inria.fr, ameny@mail.ntua.gr, {langer, wimmer}@big.tuwien.ac.at

Keywords: Software Modernization, Software Migration, Cloud Computing, Cloud Business Model, Feasibility Assessment, Reverse Engineering, Forward Engineering, Cloud Provider Benchmarking, CLOUDML@ARTIST.

Abstract: Cloud computing is still considered a disruptive technology in spite of being part of our lives for several years now. However, cloud computing is much more than a technology; it is also a business model. Many companies that have sold software in a traditional way are now attending to this revolution, wondering if that new technological and business shift is adequate for them, if they would be able to move their application towards the cloud, transforming alongside the company in a service oriented company and how they could do that. The European Project ARTIST aims to guide companies in this transition by providing them with methods, techniques, and tools, from when the migration is just a thought, until it can be provisioned as a service, taking into account technical, business and organizational aspects.

1 INTRODUCTION AND MOTIVATION

Much more than the technology that supports it, cloud computing is the latest step in the evolution of the IT industrialization process. Most CIOs wish to be on the "cloud train", and therefore cloud adoption is growing in popularity for enterprises and independent software providers. One way to quickly move to the cloud is by developing new, cloud-ready software applications that can be delivered through PaaS solutions like Google App Engine and Azure. But when cloudifying existing software applications, significant re-engineering and adaptation is needed. Only after those steps have been performed, existing applications can be delivered and offered effectively in the software-as-a-service (SaaS) model.

However, this transition can be complex, time-consuming and expensive, especially when not all applications can be moved to cloud and each application has its own business specific migration requirements that derive on the necessity of different problem approximation strategies. Questions such as *How do you know which applications are best candidates for fitting the cloud? Are there any tools that can help with application cloudification? How can enterprises take advantage of the flexibility and scalability of the cloud while avoiding application migration frustrations?* arise.

Due to these arising needs, several ongoing projects (e.g., cf. ARTIST (ARTIST, 2012), ModaClouds (MODAClouds, 2012), PaaSage (PaaSage, 2012)) explore automation possibilities for moving to the Cloud from a modeling perspective.

Besides, cloud providers' compatibility issues, vendor lock in (e.g. being non interoperable) and performance concerns still keep cloud-wary IT managers from getting more comfortable with the idea of moving applications to the cloud.

This paper proposes a global modernization framework developed in the context of the ARTIST European project which understands the cloudification of an application as a global concept involving technical, business and organizational aspects and provides methods, techniques, and tools to guide companies in this transition.

2 THE ARTIST APPROACH

2.1 ARTIST Methodology Overview

ARTIST Migration and Modernization Methodology (ARTIST WP6, 2013) has been implemented to enable the effective migration of legacy applications to cloud environments. Legacy applications have some unique characteristics which introduce many challenges for their modernization and migration to cloud environments. On one hand, there are technical issues related with the nature of the specific application and on the other, the business aspects that need to be considered in offering an application "as a service". Often the legacy applications are not cloud-enabled, following monolithic architecture design approaches and implemented in technologies which may be deprecated. The modernized version of the applications needs the equivalence of functionality and performance as well as business continuity. This does not only require adaptation of the software and integration of modern services of cloud solutions on a technical level (monitoring, security etc.), but also changing the business processes and models based on which the application is offered to the customers so as to exploit the strategic advantages of clouds.

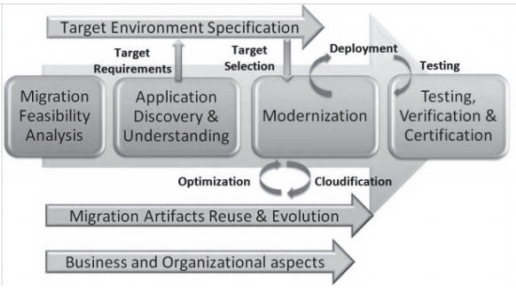

Figure 1: The ARTIST methodology overview.

The ARTIST Methodology consists of three major phases which are explained below. These phases are:

- **Pre-migration:** In this phase, a study on the technical and economic feasibility will be conducted as a prelude to perform modernization of the legacy system.
- **Migration:** This phase will perform the migration process itself, by using reverse as well as forward engineering techniques to deploy the legacy system in the cloud.
- **Post-migration:** In this phase the modernized application will be deployed on the target environment and checked if both technical and business objectives established in the pre-migration phase have been achieved. Moreover, a certification model will be created in order to increase customer confidence in the SaaS system.

2.1.1 Methodology Process Tool

The ARTIST methodology is a detailed (ARTIST WP6 2 2013), but also generic, methodology that covers all migration tasks and processes. In order to practically support this methodology, we incorporate a central component in the overall architecture: the **ARTIST Methodology Process Tool (MPT)**.

The objective of the ARTIST MPT is to allow the customization and instantiation of the ARTIST Migration Methodology for end-users based on the Migration Assessment results of the particular migration project. The MPT, exploiting the results processed and obtained during the assessment, defines a customized modernisation process, tailored to the concrete legacy application needs. The tool shows the customized process in detail, its tasks broken down step-by-step, including hooks to invoke the tools required to accomplish each task acting as an umbrella for ARTIST tools.

During the first year of the ARTIST Project, we worked on the requirements analysis to effectively design and implement MPT. Besides the requirements analysis, we also investigated the possible implementation approaches. The identified approaches are:

- Eclipse-based
- Web-based
- Spikes Together proprietary solution (SPIKES, 2013)
- Hybrid (Eclipse + Web)

After a first analysis on the pros and cons of each approach, the Hybrid solution seems to be the most adequate one. It combines the smooth integration capabilities of the Eclipse based solution (as most of the ARTIST tools will be Eclipse based) and the adaptability of the Web solution for collaborative development environments. In the next year of the project the final implementation technology will be selected and the first version of the prototype implemented.

2.1.2 Pre-migration Phase and Supporting Tools

The pre-migration takes place even before the migration starts. Software vendors need to check if

what they want to achieve with a specific migration project is actually feasible to them in terms of technology, processes and business (ARTIST D5.1.1, 2013).Once a company has realized that their current product is not sustainable as-it-is any more, the first big challenge faced is therefore the decision of what is more convenient, to migrate or to start from scratch.

ARTIST proposes a pre-migration phase that starts off with the characterization of the legacy application from two points of view: technical and business, and follows up with a technical feasibility analysis and totally intertwined, an economic and business feasibility analysis. The result of this pre-migration phase is a Go / NoGo to the migration.

The first step of the pre-migration is the **Maturity assessment** with the main objective of analysing how mature the application is in terms of technology (i.e. architecture, programming language, database, integration with 3rd party offerings, installation requirements, versioning, etc.) and business (i.e. current business model, existence of SLA, maintenance and upgrades procedures, customer service, etc.) and how the customer wants the application to be in those two axes once the application is migrated. The evaluation of the current situation and the ideal situation allows ARTIST to perform a gap analysis, described in terms of a technical feasibility analysis and the business feasibility analysis.

The Maturity Assessment in ARTIST is supported by the **Maturity Assessment Tool** (MAT). which focuses on two perspectives, business and technical in both situations (initial and final) and provides as a result a picture with the position in a quadrant of the initial and the final situation of the application enriched by a set of high level recommendations and goals to be reached along the migration project (the current prototype supports the maturity analysis, and the positioning of the application).

The second activity to be performed in the pre-migration phase is the **Feasibility Analysis which comprises a Technical Feasibility & Business Feasibility analysis.**

On the one hand the **Technical Feasibility Analysis** aims at supporting ARTIST users on the early technical assessment of the migration of a legacy application to the cloud. No matter how simple the application may be the technical feasibility analysis may require non negligible efforts and concrete expertise to be accomplished. The support for decision making at this early pre-migration stage, can benefit from a detailed breakdown of the migration process into a set of technical tasks, to estimate their required efforts, and to identify other resources needed to accomplish every task, including the selection of the appropriate technical expertise or even the detection of dependencies among tasks or other technical intricacies.

ARTIST supports the technical feasibility analysis through the **Technical Feasibility Tool** (TFT).The aim of the TFT is to estimate the efforts required to migrate a legacy application to a selected target cloud environment, fulfilling some migration goals and requirements and automating the process as much as possible. Our implementation of TFT extends the Cloud Migration Point (Tran, 2011) approach by automating some steps, using techniques such as Model Driven Reverse Engineering (MDRE), Software Metrics or DSL-based heuristics, notably to extract knowledge of the legacy system, propose migration strategies and estimate the component complexity. It provides as a final result the estimated effort required to perform the migration project (based on previous based experiences). The current available prototype supports the extraction off software metrics and the proposition of components and migration strategies.

In parallel to the Technical Feasibility Analysis a **Business Feasibility Analysis** is proposed. The need of the Business Feasibility Analysis stems the observation that, although cloud computing is a software deployment scheme expected to bring a number of advantages (e.g. elastic provisioning, cost savings) to its adopters the same advantages does not apply equally well to all potential users (ARTIST D5.1.1, 2013). Hardware costs savings, as an example, does not equally apply to SME and large enterprises. Moreover there are also potential (often sunken) costs that have to be carefully assessed. The Business Feasibility Analysis in ARTIST aims to provide, not only economic information about how ROI, or payback metrics shall behave in the future, but also which are the main risks to be faced with the migration and the organizational processes affected by the uptake of the new business model.

The tool that performs the Business Feasibility Analysis in ARTIST is the **Business Feasibility Tool** (BFT) which supports decision to estimate costs, benefits and operational changes to be applied within a migration to a cloud deployment scheme. The ARTIST's BFT adopts (i) the Agent-Based modelling (Twomey et al., 2002) paradigm to represent (cloud-based) Business Models (ii) Agent-based Simulation (Tesfatsion et al., 2006) to support

the user to learn how the organizational changes required by the adoption of a cloud deployment scheme may impact an enterprise and (iii) Enterprise Simulation (ES) (Datar et al., 2000) to support "what if" scenario aimed at the evaluation of the impact that additional controls due to the adoption of a cloud deployment scheme may have on the organization.

2.1.3 Migration Phase and Supporting Tools

2.1.3.1 Reverse Engineering

Reverse Engineering is the first step of any migration process, as it consists in performing a prior analysis of the existing system or software that is going to be later migrated (e.g. to the cloud) (Canfora et al., 2011). Within the ARTIST methodology and model-based framework, a global Model Driven Reverse Engineering (MDRE) (Rugaber et al., 2004) approach is being designed and developed to support it.

Model Discovery, cf. the **Model Discovery Toolbox (MDT)** is the initial activity of obtaining "raw" (i.e. low-level) models from the different artefacts composing a given software. This is realized (at least semi-)automatically thanks to software components called *model discoverers*. The main related challenge is to deal with the software artefacts heterogeneity, as they can take many different forms/formats (Java or C# source code, XML documents, databases, etc.). Thus, we have been working on a taxonomy of legacy artefacts that helps better classifying them according to several dimensions (their technical space, internal structure, nature, size, environment, etc). The second objective of this taxonomy is to provide some kind of guidance in the process of deciding on the most suitable way to discover the required initial models.

Model Understanding, cf. the **Model Understanding Toolbox (MUT)** is the activity of processing the initial models, computed from the Model Discovery, in order to identify and build higher-level views on the analyzed software. This is generally performed via (chains of) model transformations, usually refining the models iteratively. There are several issues related to this model understanding context: How to deal with both functional and non-functional properties of a system? How to allow the definition and computation of new views? How to generalize views independently from the nature of the system initially reverse engineered? Intending to answer these open questions, several tracks are being explored. For instance, a view definition DSL is being specified in order to make easier the elaboration of (new) views covering different aspects of a given system. Also, experimentations are being performed on which model transformations are made more generic and so reusable indifferently in the context of several types of systems (e.g. both Java- and C#-based).

These two toolboxes come up with sets of components that can be picked up, used and combined according to the context (i.e. the input system or software, the targeted migration platform, etc.). The final goal is to be able to feed correctly the next steps of the process, and more particularly the beginning of the Forward Engineering phase.

2.1.3.2 Target Platform Selection

One of the most complex activities when migrating to a cloud based solution is the selection of the appropriate cloud platform. Here, several aspects come to play, such as the different services offered by the cloud platforms (in any of their delivery models IaaS, PaaS or SaaS) and the needs from the migrated software to be deployed in an efficient manner (in terms of costs) and fulfilling the required QoS parameters (without compromising the SLAs agreed by the service provider and the final user) in the selected provider.

To ease this process ARTIST proposes (Menychtas et al., 2013) to implement two phases in parallel: one in the application domain and one in the candidate target environment domain.

In the application domain, following the analysis of the application features and the creation of models describing the legacy system using reverse engineering techniques, we examine and profile the performance aspects of each application element and feature (through the profiling tool). In the process, these aspects are linked with specific software solutions exploiting trace analysis and benchmarking, which in sequel are matched to elementary hardware resources (virtualized or physical) such as computation storage and networking.

In the target environment domain, the various offerings, in the IaaS, PaaS and SaaS layers, are described in a way that will facilitate the matching between these offerings and application component requirements. For this, we have developed a unique Cloud Modeling Language (CloudML@artist). The specification of the target environment, including both functional capabilities and detailed performance aspects of common application types, enables effective matchmaking and allows for the selection of the ideal provider for the overall application and/or its specific fragments. Metrics are

also investigated to characterize service offerings based on a combination of performance and cost, to improve the overall deployed application's efficiency.

Models of the candidate cloud targets (infrastructure or platform providers), which are available through the ARTIST repository are matched against the afore-posed requirements, using a model-enabled matchmaking algorithm. The optimal target provider (e.g. best ranked match) could be automatically selected or the selection can be deferred to the end-user, who selects it amongst the found matches.

2.1.3.3 Forward Engineering

The main goal of the **forward engineering phase** is to produce executable code, which is efficiently runnable in cloud environments, from platform independent models (PIMs) which have been produced in the reverse engineering phase. In order to produce the required code from the given models, we use several dedicated steps to cope with the following main challenges. First, the PIMs have to be refined in order to deal with the specifics of a particular cloud environment–this step is called *model cloudification*. For instance, a first basic step when migrating models towards cloud environments is to select appropriate cloud services and a virtualization level. Second, possible *model optimizations* have to be explored in the refinement process to meet the non-functional requirements stated for the migration. Current cloud environments offer a large set of different configuration options. Thus, automation support is urgently required to explore more efficiently and effectively this search space. Third, tailored code for a particular cloud environment has to be generated from the refined models, packaged for, and deployed in the chosen cloud environment involving also the provisioning of the required cloud resources beforehand.

Our approach to support this phase relies on **model transformations as main means for automation**. In particular, we employ different model refinement levels instead of generating code directly out of the PIMs. Using these subsequent refinement steps, we ensure that developers are able to control and guide the model cloudification and optimization for their purposes. We also aim at eliminating tedious recurring tasks by applying the convention-over-configuration principle as well as design exploration techniques when transitioning from PIMs to cloudified platform-specific models (PSMs).

As the Unified Modeling Language (UML) (OMG, 2011) already provides modelling concepts to represent software-, platform-, and infrastructure-related artefacts, especially in version 2.x, e.g., considers UML deployment diagrams, UML is used as host language for specifying PIMs and PSMs in our approach. UML is designed as a platform-independent and general purpose modeling language, but also as an extensible language for considering specific domains and technologies such as cloud computing. The language inherent extension mechanisms of UML are *UML libraries* and *UML Profiles*. On this basis, we developed as one part of the CloudML@ARTIST language a sublanguage called *Cloud Application Modeling Language* (CAML, available as open source project at: http://code.google.com/a/eclipselabs.org/p/caml), which provides cloud-specific modeling concepts and at the same time well established UML modeling concepts. In this respect, we have extended UML by providing the UML-based *Cloud Deployment Modeling Library*. This library allows representing cloud-based deployment models independent from particular cloud providers. To keep the cloud provider specifics separate from the modeling library, additional UML Profiles are utilized. These UML Profiles are essential to allow cloud consumers specifying concrete deployments for a selected cloud provider. In the current first version of CAML, we developed UML Profiles that address functional cloud consumer concerns, such as instance types, storage solutions and service offerings, as well as non-functional ones, such pricing, performance and service levels.

To transition from PIMs to cloudified PSMs, we provide a set of **model transformations** that achieve the refinement of the models based on cloud blueprints, patterns, and best practices. Based on the stated migration goals, the abstract services contained in the PIMs are substituted by the most appropriate specific ones offered by cloud providers. Once a set of services is selected, additional configurations and deployment options are automatically examined in order to improve, e.g., the performance measures or costs of software running in the cloud. Therefore, we reuse **simulation techniques** for the PSMs that are established for the migration validation in the post migration phase (cf. Subsection 3.1.4.2). Based on the results, we foresee dynamic design-space exploration by applying different cloud optimization patterns formalized as model transformations to the models and evaluate their impact on the non-functional properties by iterative simulation runs (Troya et al., 2013).

From the optimized PSMs, application code is

generated using **model-to-text transformation** approaches. Please note that not only the application code is produced, but also supporting code, e.g., deployment scripts, configuration files, etc., are generated. These additional artefacts are needed to automatically transfer the generated applications to pre-configured cloud environments.

We aim for a **highly configurable forward engineering process** built up of reusable transformations (Kusel et al., 2013). By using different model composition techniques to develop transformations chains, we counteract the development of monolithic transformations. By reusing standards, such as UML, and Ecore, we have several possibilities to reuse available transformations for code generation and available metamodels and UML profiles that may be of interest also for the cloud modeling domain such as SoaML and MARTE.

2.1.3.4 Business Model Definition and Organizational Impact

Most of the existing cloud migration approaches (ARTIST DOW, 2012) focus their attention in the analysis and the implementation of the technical migration strategy. But cloudification paradigm affects also the delivery model of the company (from SaaG to SaaS) which implies change also at organizational and business level.

Within ARTIST methodology we propose, in combination with the technical migration activities, a set of tasks to implement the changes required as a result of the migration at business and organizational level.

The activities required **for the re-definition of the business model** are based on Osterwalder's Business Model Generation and adopted for cloud based applications, including Market segment re-definition, cost-structure re-development or customer relationship update among others.

The tasks related to the **organizational changes** include aspects from development process to accountability or providers management processes specially adapted for Cloud based and SaaS provider companies. These ideal processes are based on best practices and standards such as CCRA (ITU-T SG13, 2013).

2.1.4 Post-migration Phase and Supporting Tools

2.1.4.1 Migration Goals

The main objective of the migration goals is to establish in a formal and shareable manner the main **constraints exposed from the application (or its owner) for the migration**. The requirements for the migration that cannot be extracted from the legacy software itself will be established by the migration goals and circulated over the other ARTIST tool to be accessible in all migration phases.

These migration goals are worked out through a questionnaire and finally determined by the end-user and are circulated over other migration phases with the purpose of having information about the non-functional requirements required for the migration (quality metrics required, infrastructure information, etc.). These user specified migration goals will be validated in combination with the functional requirements in the validation phase c.f. subsection 3.1.4.2.

2.1.4.2 Validation of the Migration

The objective of the validation phase is to assess whether the migration was successful or not. A migration is considered to be successful, if the b**ehaviour of the software** has not been affected in an unexpected way during the migration (i.e., preserving functional correctness) and if the **migrated software fulfils the user-specified migration goals**.

Validating these two aspects holds several challenges. For evaluating the functional correctness of the software after the migration, it is necessary to verify whether the business logic of the software after the migration corresponds to the software's business logics before the migration. Especially when an insufficient number of test cases is available for the software, ensuring the correct behaviour of the migrated software with respect to the behaviour before the migration is a challenging task—from a theoretical and a practical perspective. **Concerning the validation of the user-specified migration goals**, we have to evaluate a variety of different types of potentially competing migration goals, ranging from goals, for instance, regarding performance efficiency to goals concerning operating costs. These goals may often not be evaluated by measuring the respective metrics directly in the software deployed in the cloud environment used in production, because, for instance, benchmarking the deployed software in the cloud environment may lead to high costs or the information required for the evaluation may not be available directly due to restrictions and virtualization levels of the cloud environment.

For assessing the functional correctness of the migrated software, we follow three different

approaches. First, we adapt the migration techniques of ARTIST in order to migrate not only the software itself but also potentially existing test cases. If no test cases are available or if their test coverage of is insufficient, we follow a second approach to assess the functional correctness. In particular, we provide techniques to generate test cases automatically from the PIM, corresponding to the overall spirit in ARTIST to apply techniques from model-driven engineering. Therefore, not only structural but also behavioural models are reverse-engineered from the legacy software and represented in the PIM. These behavioural models are then used in a model-based test generator tool developed in the course of ARTIST to derive test inputs until the execution of the software with these test inputs leads to a certain test coverage. For this task, we adapt differential symbolic execution techniques to be applied solely on model-level (Person et al., 2008). Of course, this step has to be guided by the user, who may indicate which parts and to which degree the migrated software shall be tested. If the behavioural models are detailed enough, we may also use them to derive test oracles for the generated test inputs. First experiments show that FUML (OMG, 2012), a recent standard of the OMG providing a concise semantics definition for a subset of UML Classes and Activities, may serve as an adequate modelling language to represent the expected behaviour of the software accurately and independently of any platform or technology. In fact, the semantics of FUML is precise enough to enable the unambiguous simulation of FUML models. Therefore, we developed an extended FUML virtual machine (Mayerhofer et al., 2012), which enables the simulation of FUML models, while providing detailed information about the simulation, such as inputs and outputs of each step and detailed simulation traces. Based on this information, we simulate the models for each test input and obtain test oracles covering the expected outputs, as well as the expected traces. The third approach to assess the functional correctness of the migrated software is based on running the legacy software and the migrated software in parallel, forward each request to both of software versions, and compare their responses.

To provide the means for evaluating whether the migration goals, such as performance efficiency and operation costs, are fulfilled, we have to address the challenge that certain information for computing the verdict about the migration goal may not be available in the cloud environment used for production. Thus, simply benchmarking the deployed software may sometimes be insufficient. Therefore, we provide a dedicated tool that enables model-simulation and analysis techniques to estimate certain property measures that are related to the respective quality characteristics mentioned in the migration goals. These measures are then combined and analyzed to validate whether the defined goals are fulfilled. Our current work shows promising first results indicating that several metrics can be obtained by model-level simulation leveraging the level of details in the PSMs (representing the migrated software on model level) without requiring a translation of the PSMs into dedicated performance models (Berardinelli et al., 2013); (Fleck et al., 2013) which is however the case for existing work in software performance analysis (Balsamo et al., 2004). These techniques come with the additional benefit that they can be performed also when the actual migrated code has not been derived yet and hence may be used to guide the optimization already in early phases of the migration as well as in the forward engineering phase (cf. Subsection 3.1.3.3). However, since the model-level simulations and analyses are only estimations, we further aim at validating the migration goals on code-level as a final step. We run and benchmark the actual migrated software, as far as possible (van Hoorn, 2012) providing dedicated forward-engineering modes that, besides generating the production code, also instrument the generated code, where necessary, to obtain the measures needed to validate the migration goals.

2.1.4.3 Certification Model

One of the shortcomings for the final implantation of SaaS and cloud computing in the software industry is the reluctance of the users to this new service offering.

The purpose of the certification phase in ARTIST is to obtain an independent and impartial judgment of SaaS providers, focusing on businesses, processes and technology aspects (ARTIST D11.4.1, 2013) in order to create consumer confidence in software applications offered as services considering reliability parameters (categories) that will be evaluated.

The certification model proposed by ARTIST, Service based Software Provider (SbSp) focus on organisations that develop and offer software based services using methodologies and business models that are connected to the Future Internet and cloud computing schemes. It is structured into three areas i) The business area aims to analyse the financial

stability and soundness of the business in order to assess the potential for continuity and sustainability, ii) the process area aims to ensure the quality of the process of delivering the IT service to the end customer, and iii) the technology area aims to establish a high level of security and transparency for both customers and providers of dedicated SaaS.

To support the certification process, three questionnaires (corresponding to the three areas) have been developed to be used in the certification method. This method is the procedure through which a third independent part evaluates the practices of a company providing a cloud application to assure by a label that the system fulfils a specified level (Gold, Silver or Bronze). It is a way to ensure that "standard-based" products are implemented: quality products, competitive markets with more choices, commodity pricing, and less opportunity to become "locked in" to a particular vendor.

3 CONCLUSIONS

In this paper we present a novel cloudification approach, supporting software vendors paving the path to a smoother transition to the cloud computing paradigm. The presented ARTIST methodology and framework considers both technical and business aspects of the legacy applications and covers not only the core migration phase but also the pre-migration phase, where based on the assessment of the initial and target situations added to the results of a technical and business feasibility analysis, the various steps and tasks of the methodology are customized, as well as the post-migration phase where the outcomes are validated and certified.

ARTIST proposes a set of tools supporting each methodology phase based on model-driven engineering approaches, allowing when possible the reuse of artefacts. The first version of the ARTIST approach has been applied to the PetStore example as the preliminary test case for validation. Currently, the solution is applied to 4 real-world business scenarios for validation and refinement of the overall approach, improving its impact and standardization on real-world industrial environments.

ACKNOWLEDGEMENTS

This work has been supported by the ARTIST Project and has been partly funded by the European Commission under the Seventh (FP7 - 2007-2013) Framework Programme for Research and Technological Development, grant no. 317859.

REFERENCES

OMG, 2011. *OMG Unified Modeling Language (OMG UML), Superstructure 2.4.1.* Object Management Group, http://www.omg.org/spec/UML.

Kusel, A., Schönböck, J., Wimmer, M., Kappel, G., Retschitzegger, W., Schwinger, W., 2013. *Reuse in Model-to-Model Transformation Languages: Are we there yet?*. SoSym, Springer online first, pp. 1-31.

ARTIST D6.2.1, Menychtas, A., Orue-Echevarria, L., 2013. *D6.2.1 ARTIST Migration Methodology*. http://www.artist-project.eu/sites/default/files/D6.2.1_ARTIST%20Methodology_M12_30092013.pdf

ARTIST D6.3.1, Menychtas, A., 2013. D6.3.1 ARTIST *Methodology Process Framework*. http://www.artist-project.eu/sites/default/files/D6.3.1%20ARTIST%20Methodology%20process%20framework_M12_30092013.pdf

ARTIST D5.1.1, Alonso, J., D5.1.1. *Specification of the Business and Technical Modernization assessment in ARTIST.*

Tran, V.T.K., Lee, K., Fekete, K., Liu, A., Keung, J., 2011. *Size Estimation of Cloud Migration Projects with Cloud Migration Point (CMP)*. Int. Symposium on Empirical Software Engineering and Measurement (ESEM'11), pp 265-274.

Twomey P., Cadman, R., 2002. *Agent-Based Modelling of Customer Behavior in the Telecoms and Media Markets*, Info, Vol. 4(1), pp. 56-63.

Tesfatsion, L., 2006. *Agent-Based Computational Economics: A Constructive Approach to Economic Theory*. Handbook of Computational Economics, Vol. 2, North-Holland/Elsevier.

Datar, M.M., 2000. *Enterprise Simulations: Framework for a strategic application*. Winter Simulation Conference.

Menychtas, A., Santzaridou, C., Kousiouris, G., Varvarigou, T., Gorronogoitia, J., Strauss, O., Senkova, T., Orue-Echevarria, L., Alonso J., Bruneliere, H. Pellens, B., Stuer, P., 2013. *ARTIST Methodology and Framework: A novel approach for the migration of legacy software on the Cloud*. MICAS 2013.

ARTIST DOW, 2013. DOW: *Advanced software-based service provisioning and migration of legacy software*.

ARTIST D11.4.1, Vergara, M., 2013. *ARTIST SbSp Certification Model.*

Berardinelli, L., Langer, P., Mayerhofer, T., 2013. *Combining fUML and Profiles for Non-Functional Analysis Based on Model Execution Traces. Qosa'13, ACM Sigsoft Conference on the Quality of Software Architectures*. ACM, pp. 79-88.

Fleck, M., Berardinelli, L., Langer, P., Mayerhofer, T., Cortellessa, V., 2013. *Resource Contention Analysis of*

Service-Based Systems through fUML-Driven Model Execution. Int'l Workshop Non-functional Properties in Modeling, CEUR, Vol-1074 pp. 6-15.

Mayerhofer, T., Langer, P., Kappel, G., 2012. *A runtime model for fUML*. Int'l Workshop on models@run.time, ACM, pp. 53-58.

Person, Suzette, et al. "*Differential symbolic execution.*". 16th ACM SIGSOFT International Symposium on Foundations of software engineering. ACM, 2008.

Balsamo, Simonetta, et al. "*Model-based performance prediction in software development: A survey.*" Software Engineering, IEEE Transactions on 30.5 (2004): 295-310.

Van Hoorn, André, Jan Waller, and Wilhelm Hasselbring. *"Kieker: A framework for application performance monitoring and dynamic software analysis."* Joint WOSP/SIPEW international conference on Performance Engineering.

ACM, 2012.OMG, 2012. *Semantics of a Foundational Subset for Executable UML Models (FUML)*. Object Management Group http://www.omg.org/spec/FUML

SPIKES, Spikes is member of ARTIST consortium

G. Canfora, M. Di Penta, L. Cerulo, *Achievements and Challenges in Software Reverse Engineering*. ACM 54 (2011) 142-151

S. Rugaber, K. Stirewalt, *Model Driven Reverse Engineering*, IEEE Software 21 (2004) 45-53.

CCRA, *Cloud Computing Reference Architecture* CT-CCA-o-016-Edit-Draft-RA ITU-T SG13/WP6.

Troya, J., Cubo, J., Martín, J.A., Pimentel, E., Vallecillo, A., 2013: *Automated Throughput Optimization of Cloud Services via Model-driven Adaptation*. MODELSWARD, pp. 356-362

ARTIST 2012, www.artist-project.eu

ModaClouds 2012 http://www.modaclouds.eu

PaaSage 2012 http://www.paasage.eu