Software modernization and cloudification using the ARTIST migration methodology and framework

Andreas Menychtas, Kleopatra Konstanteli, Juncal Alonso, Leire Orue-Echevarria, Jesus Gorronogoitia, George Kousiouris, Christina Santzaridou, Hugo Bruneliere, Bram Pellens, Peter Stuer, et al.

DOI 10.12694/scpe.v15i2.980
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 15, Number 2, pp. 131–152. http://www.scpe.org

SOFTWARE MODERNIZATION AND CLOUDIFICATION USING THE ARTIST MIGRATION METHODOLOGY AND FRAMEWORK

ANDREAS MENYCHTAS[1], KLEOPATRA KONSTANTELI[2], JUNCAL ALONSO[3], LEIRE ORUE-ECHEVARRIA[4], JESUS GORRONOGOITIA[5], GEORGE KOUSIOURIS[6], CHRISTINA SANTZARIDOU[7], HUGO BRUNELIERE[8], BRAM PELLENS[9], PETER STUER[10], OLIVER STRAUSS[11], TATIANA SENKOVA[12], AND THEODORA VARVARIGOU[13]

Abstract. Cloud computing has leveraged new software development and provisioning approaches by changing the way computing, storage and networking resources are purchased and consumed. The variety of cloud offerings on both technical and business level has considerably advanced the development process and established new business models and value chains for applications and services. However, the modernization and cloudification of legacy software so as to be offered as a service still encounters many challenges. In this work, we present a complete methodology and a methodology instantiation framework for the effective migration of legacy software to modern cloud environments.

Key words: Cloud Computing, Legacy Software, Modelling, Migration, Methodology, SaaS

AMS subject classifications. 68M14, 68U01, 68U35, 90B25

1. Introduction. Nowadays, cloud computing [4] appears as one of the most popular and mature technological and business environments for engineering, hosting and provisioning software applications. A continuously increasing set of cloud-based solutions across the cloud stack layers [11] is available to application owners and developers to tailor their applications and exploit the advanced features of this paradigm for elasticity, high availability and performance. These solutions provide many benefits to new applications but they also introduce constrains to the modernization and migration of legacy applications. We consider *legacy applications* as software not developed for the Cloud and software in traditional architectural paradigms that cannot be scaled, cannot be measured and does not share resources beyond infrastructure (e.g. database, memory). Often the legacy applications follow monolithic architecture design approaches, implemented in technologies which may be deprecated or cannot easily deal with the notion of "as a Service" and are installed on owned infrastructures.

The modernization and adaptation of legacy applications to cloud environments is a great challenge for all involved stakeholders, not only from a technical perspective, but also in business level with the need for adaptation of the business processes and models of the application which will be deployed on the Cloud and offered "as a service". In this paper, we present a novel model-driven [22] approach for the migration of legacy applications in modern cloud environments which covers all aspects and phases of the migration process, as well as an integrated framework that supports all migration process.

Our motivation for this work is the requirements and challenges for the effective migration of legacy software on the Cloud as described in [9]. To this end, the proposed migration methodology considers the following aspects:

- **Unknown internal structure** due to the complexity of the software and the data management processes.
- **Lack of knowledge for target environment** where the application will be deployed and provisioned.

[1]National Technical University of Athens (ameny@mail.ntua.gr)
[2]National Technical University of Athens (kkonst@mail.ntua.gr)
[3]Tecnalia Research & Innovation (juncal.alonso@tecnalia.com)
[4]Tecnalia Research & Innovation (leire.orue-echevarria@tecnalia.com)
[5]ATOS Research & Innovation (jesus.gorronogoitia@atosresearch.eu)
[6]National Technical University of Athens (gkousiou@mail.ntua.gr)
[7]National Technical University of Athens (csantz@mail.ntua.gr)
[8]Inria, Mines Nantes & LINA (hugo.bruneliere@inria.fr)
[9]Spikes N.V. (bram.pellens@spikes.be)
[10]Spikes N.V. (peter.stuer@spikes.be)
[11]Fraunhofer Institute for Industrial Engineering IAO (oliver.strauss@iao.fraunhofer.de)
[12]Fraunhofer Institute for Industrial Engineering IAO (tatiana.senkova@iao.fraunhofer.de)
[13]National Technical University of Athens (dora@telecom.ntua.gr)

- **Multi-tenancy influence** which creates a number of issues that span from security to performance and availability.
- **Variable configuration based on user preferences** which poses additional customization requirements when the application is offered as a service.
- **Various application types** with different characteristics and usage of resources e.g. networking, storage etc.

Unlike current methodologies, in this work we present an end-to-end approach developed in frame of ARTIST EU Project [1] that covers both all migration and modernization phases and also considers both the technical and business aspects of the application. This complete model-driven modernization and migration approach (initially discussed in [12]) is supported by an innovative toolbox as a "one stop shop" for the migration, modernization and cloudification of legacy applications. ARTIST Migration Methodology introduces three phases for the migration of legacy applications to cloud environments starting from a feasibility analysis to the core modernization of the software and its validation and certification during the post-migration. All aspects of the migrated software are examined in the methodology including the migration of the data and the adaptors which may be required for the handover between the legacy data source and the modern cloud data stores.

In addition, the proposed framework includes a rich set of tools, which realize each phase, task and activity of the methodology. The methodology and the framework provide a complete environment where all the stakeholders involved in the migration of an application (analysts, developers, engineers etc.) can collaborate under well defined workflows and processes that are customized and instantiated for the particular migration project. The post-migration aspects are also considered by defining specific processes for the behavioural equivalence (functional and non-functional parameters) and the reuse of software artefacts.

The rest of the paper is structured as follows: Section 2 highlights the related work in this field while Section 3 describes in detail the proposed migration methodology. Section 4 analyses the architecture design of the overall framework that supports the methodology and Section 5 the implementation details of the Methodology Process Tool, which is the core component for the customization and instantiation of the methodology. An evaluation of the proposed solution based on an experimental application is described in Section 6. Finally in Section 7, the conclusions of our work are presented.

2. Related Work. Prior to the definition of the various methodology elements, we examined the related migration approaches and extracted some interesting findings, with the most important one the fact that there is no methodology covering all migration processes/phases required in our approach.

G. Lewis et al. [10] proposed the SMART methodology which is relevant to the pre-migration phase while trying to understand the system operation in order to be able to identify and analyse the gap between the legacy system and the desired one. However, most of this analysis is performed manually and based on the knowledge of the participating team. After acquiring that knowledge, SMART proposes an ad-hoc migration strategy created for each system individually. In addition, since SMART is focused on the migration to SOA, many relevant issues that concern the basics of SaaS architectural design are not treated.

The Butterfly method [26] guides the migration of a mission-critical legacy system to a target system and consists of 5 phases, namely: *justification, legacy system understanding, target system understanding, migration* and *testing*. Justification phase involves the investigation of the risk and benefits associated with the legacy system evolution while legacy system understanding involves reverse engineering of the legacy system in order to identify the components, recreate documentation, understand the static and dynamic behaviour of the legacy system. Target system development involves elicitation of requirements/specifications of the target system and choosing the most appropriate architecture and standards. The migration phase is concerned with the physical transformation of the whole legacy system to the target system.Finally, testing is carried out throughout the evolution process to ensure that the target system delivers the functionalities specified at the start of the evolution.

Warren and Ransom [25] developed the Renaissance method which examines a legacy system from all technical, business and organizational perspectives. The method guides users through assessment of these perspectives by selecting assessment characteristics and assigning values to them. According to this approach system assessment is used to gain an understanding of a legacy system, which is fundamental to any system evolution exercise. System assessment should be an initial activity for evolution projects. The first important

milestone in the Renaissance method is a viable, cost effective system evolution strategy which can be presented to higher management. The activity of assessing the current system is supported by another activity for modelling which is focused on increasing the level of knowledge about the system on a conceptual level.

OMG ADM (Architecture-Driven Modernization) [16] provides a set of generic metamodel specifications that could be relevant within the context of our work. The Software Measurement Metamodel (SMM) proposed by this methodology, could be used in the assessment of the migration process (e.g. for expressing appropriate metrics/measures as well as representing the result of their computation), the Knowledge Discovery Metamodel (KDM) and to a lower extent the Abstract Syntax Tree Metamodel (ASTM) can be exploited for reverse engineering purposes (e.g. for representing the legacy source code as models in a neutral technology-independent manner).

A. Bagnato et al. [2] implemented the XIRUP, which is considered as a general-purpose MDE-based modernization methodology, not specifically designed to address the challenges of migrating legacy applications to cloud environments. In particular, XIRUP is a feature-driven modernization methodology, where the whole legacy system is decomposed into features (as offered by encapsulated components) that are iteratively evaluated (for migration decision support), migrated and assessed (post-migration evaluation). Nonetheless, even if some of these method fragments are applicable to foreseen methodology phases, they need to be aligned and extended to cope with the particularities of the migration to the Cloud.

Model Driven Reverse Engineering (MDRE) itself, as the application of Model Driven Engineering (MDE) principles and techniques to reverse engineering problems, is a rather recent area [21]. However, there have been really few generic and extensible MDRE approaches proposed so far. The MoDisco two-step Model Discovery + Model Understanding approach and corresponding framework in Eclipse [3] is one of them. As covering the reverse engineering phase, it can be directly reused and then extended when necessary within the context of the more global approach.

The REMICS methodology proposed by Mohagheghi and Sæther [15], while following an MDE approach, does not take into consideration non-functional requirements (i.e. performance) inherent to SaaS applications and neither addresses architectural issues such as multi-tenancy or scalability. REMICS relays the monitoring, billing and security issues to the cloud provider where the application is deployed on. The business issues (business model and processes), related to the components mentioned before (billing, monitoring) are also not considered. In addition, REMICS executes the migration on brute force, without considering the feasibility or convenience to migrate.

In the following table we summarize the various aspects of the migration methodologies defining the baseline for the ARTIST Migration Methodology:

TABLE 2.1
Migration Methodologies Comparison

	Business Aspects	Technical Aspects	Cloud Aspects	Migration Assessment	Validation & Verification
SMART	✘	✔	✘	✔	✔
Butterfly	✔	✔	✘	✔	✔
Renaissance	✔	✔	✘	✔	✘
OMG ADM	✘	✔	✘	✔	✘
XIRUP	✔	✔	✘	✔	✔
MoDisco	✘	✔	✘	✘	✔
REMICS	✘	✔	✔	✘	✔

3. ARIST Migration Methodology.

3.1. Methodology Overview. The methodology was developed to cover all aspects of software migration and modernization, including the business and technical requirements posed from the modern cloud ecosystems as common target environments for the deployment and provisioning of applications. The methodology consists of three (plus one) major phases which are analysed below:

These three main phases are:

- **Pre-migration:** In this phase a study of the technical and economic feasibility will be conducted as a prerequisite to the migration/modernization of the legacy system so as to effectively define the exact steps and effort that will be required.
- **Migration and Modernization:** This phase will perform the migration process itself, by using both reverse engineering (RE) and forward engineering (FE) techniques in order to deploy the legacy system on the Cloud. It should be noted that based on the particular legacy application requirements and the pre-migration analysis results, the migration processes and their flow are explicitly customized. The business model concerns are also studied and defined in this phase. These business issues are included in the application architecture and organizational processes to face the new situation are also (re)defined in this phase. Finally, migration phase includes the verification and validation (V & V) of the final system.
- **Post-migration:** In this phase, the modernized application components will be deployed onto the target environment and it will be checked if both the technical and business objectives established in the pre-migration phase have been achieved. The validation activities that are foreseen are mainly focused on behavioural equivalence, model based testing and end-user functional and non-functional testing. Moreover, a certification model will be created in order to increase customer confidence in the SaaS system.

In addition to the three phases described above, another phase, named **Migration Artefacts Reuse and Evolution**, was defined in order to enable the effective reuse of software artefacts and optimization of the migration processes. This phase includes all needed application maintenance activities after migration to the Cloud (software updates, cloud provider changes, etc.).

The complete methodology is formally described using the Eclipse Process Framework EPF [7] and SPEM2.0 [17] specification and the generated diagrams of each phase are presented in the following sections.

3.2. Pre-Migration Phase. The pre-migration phase (Figure 3.1) is the starting point of each migration. Migration of legacy applications is considered as a very challenging project that involves not only changing the way companies are going to deliver their software, but also their business model, and how the company is organized in terms of processes. Thus, software vendors need to analyse whether the targeted objectives are actually feasible to them both technologically and economically.

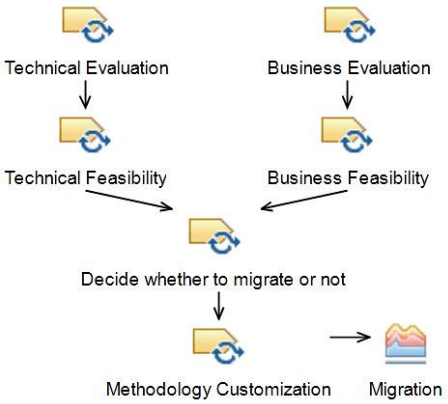

FIG. 3.1. *Pre-migration EPF Model*

The first step in this pre-migration phase is to analyse how mature the application is in terms of technology (i.e. architecture, programming language, database, integration with third party offerings, etc.) and business (i.e. current business model, maintenance and upgrades procedures, etc.). It also considers how the customer wants the application to be in those two axes (i.e. architectural design, cloud provider requirements, business model, legal concerns, performance thresholds values, etc.) once the application is migrated. The analysis of

both the current and ideal situations allows the users to perform a gap analysis, described in terms of a technical feasibility analysis and a business feasibility analysis.

The technical feasibility analysis is aimed to provide a snapshot of the application's design quality, of its complexity and coupling, etc. This is realized thanks to reverse engineering techniques through which the component model of the source code is obtained and an evaluation of the effort required to perform the migration is implemented. The technical feasibility analysis provides a set of migration strategies for each of the components identified in the legacy application and the effort required to perform that strategy. This information is obtained by the combination of the the ideal technological maturity identified in the maturity assessment as well as the target platform requirements, the complexity due to the migration strategy nature and the complexity inherent to source code (the latter is calculated performing a static analysis of the source code).

Furthermore, based on the results from the ideal situation identified in the maturity assessment and the identification of the target platform expected characteristics, a business feasibility analysis is performed. This business feasibility analysis aims at providing not only economic information (ROI, payback, etc.) but also what are the main risks to be faced with the migration and the organizational processes affected by the uptake of the new business model. The results obtained will guide decision makers to identify the most appropriate strategy in terms of migration and selection of the target services/platforms to use.

3.3. Migration and Modernization Phase. The Migration phase is the core part of our methodology, incorporating unique features for reverse and forwards engineering as well as for the effective target environment specification.

3.3.1. Application Discovery and Understanding. During the technical feasibility assessment as well as during the migration process itself, the discovery of relevant models describing the software system is first required in order to 1) have a better understanding of it and 2) provision the other steps of the process with the needed models. Thus, one of the main goals of Model Driven Reverse Engineering (MDRE) is to extract the overall structure and logic of the software system (as well as some more implementation details when necessary for the actual migration), considering different abstraction levels depending on the targeted stakeholders and migration scenarios. In any case, such a MDRE process is generally composed of the two main tasks (Figure 3.2).

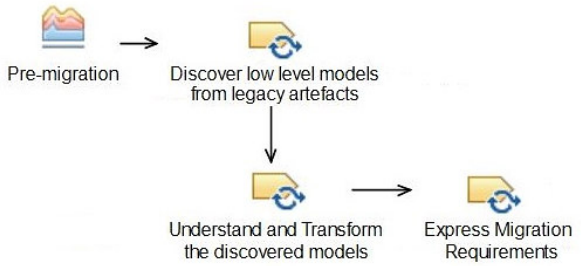

Fig. 3.2. *Application Discovery and Understanding EPF Model*

Model Discovery generates the minimum set of required initial "raw" (i.e. low-level) models out of (some of) the artefacts composing the software system. This is realized (at least semi-)automatically thanks to dedicated software components called model discoverers. This first task notably implies analysis the different available software artefacts to identify the ones which are actually usable and relevant to the considered migration scenario, and also to guide the discovery process itself. This preliminary action can be performed with the help of the *Taxonomy of Legacy Artefacts*, as established and developed within the time-frame of the ARTIST Project, which allows for better classification of these artefacts according to several common dimensions (their technical space, internal structure, nature, size, environment, etc).

Model Understanding produces the necessary "processed" derived models from the initially discovered models, thus filtering only the information required for the remainder of the overall process (i.e. Modernization

notably). This is realized in order to identify and build higher-level views on the analysed software system. This second task relies on the use of various (chains of) model-to-model transformations from the previously obtained "raw" (low-level) models to the final derived models to be provided as useful views and/or inputs of the Modernization tasks (notably). These derived models may conform to different metamodels and so offer several "viewpoints" on the software system at different levels of abstraction, according to the actual requirements of the next tasks and corresponding architects/engineers.

3.3.2. Target Environment Specification. In the Target Environment Specification phase (Figure 3.3) we have two independent processes taking place, one in the application domain and one in the candidate target environment domains, that can be linked at the end for added value. In the first place, Target Environment Benchmarking is performed in order to acquire measurements of cloud services performance in a variety of different application types (e.g. DBs, server based apps, Java based apps etc.). This information is then included in concrete provider models that contain also numerous functional characteristics. This is a process that is performed "offline" and by a suitable role (e.g. Benchmarks provider). Furthermore, in these models we include also cost information, that can be used together with performance aspects, in order to rank services based on their combined metrics, supporting also different ratings based on user interests that are expressed through percentages (e.g. 70% interest on performance, 30% on cost). More information on this process can be found in [8].

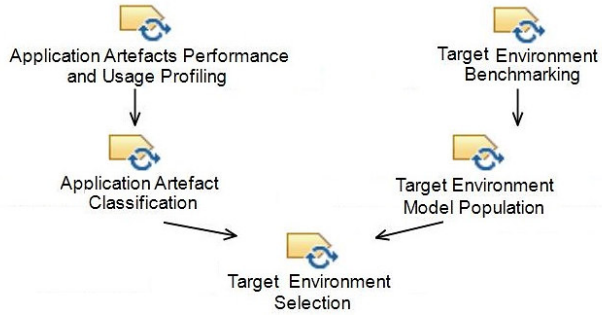

FIG. 3.3. *Target Environment Specification EPF Model*

The second branch of the process is related to the analysis of the performance footprint of arbitrary application components. In this process, which is performed by the Application Developer through the use of the respective tools, the application component's runtime behaviour is compared and classified based on the behaviour of the benchmarks used in the Target Environment Benchmarking, when both are executed on the same reference platform (of the Application Developer). In this process, the developer utilizes the *Benchmarking Tool* (to install and execute the benchmarks locally) and the *Profiling Tool* (to acquire the execution footprint of both the benchmarks and the application component). The final step is to feed this information in the *Classification Tool*, that will perform the matchmaking. Thus, after having this classification conclusion, we can select the cloud service that has the best score in the same benchmark category that the application component has been classified. It is necessary to stress that this part of the process is optional for the developer. If they have already knowledge of their application behaviour or nature, they can directly ask a question of the form "Give me the best offering for running my (web server) application", along with the aforementioned percentages of interest described in the first phase.

3.3.3. Modernization. During the migration phase, once the legacy system has been well understood and decomposed into features and/or components, other tasks such as Model Driven Forward Engineering (MDFE) need to be applied aiming at transforming the legacy system and deploying the migrated one into the target Cloud. During the understanding phase, some high abstract level model views of the legacy system have been produced. These views represent, e.g. the platform independent models (PIM) of the legacy system. Feature or component views are useful since they enable a feature-driven iterative migration across a number of selected features or components. For instance, persistence could be a feature to iterate on: during this iteration all data

sources marked for migration are transformed.

In this modernization phase (Figure 3.4), some artefacts produced during early phases are taken as input: a) the PIM models of legacy system, b) the architectural constraints (e.g. migration goals) corresponding to the target maturity level the application and c) the models describing existing target platforms where the application can be deployed.

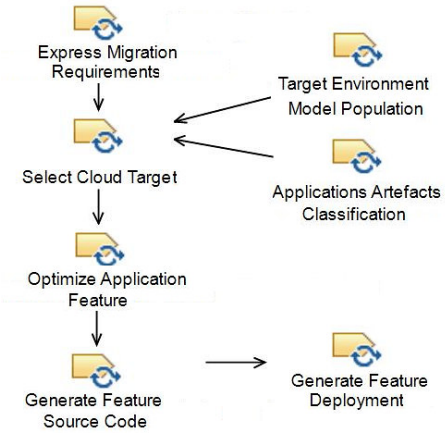

FIG. 3.4. *Modernization EPF Model*

Models of the candidate cloud targets environments are matched against the aforementioned requirements, using a model-enabled matchmaking algorithm. Models describing features or components tagged for migration are optimized and transformed, by applying suitable transformation patterns, into target models. These models represent equivalent features or components that are compatible with the selected target provider and that fully exploit the target features and services requested by the application requirements.

These migration tasks aim at building and deploying the migrated component corresponding to selected legacy components (e.g. implementations of application features). MDFE strongly relies on models and model transformations e.g. model to model (M2M) and model to text (M2T) for the technical materialization of the described MDFE tasks. The main aim for these transformations is to generate diverse models providing different views over the migrated application, including its source code. The skeleton of the application is generated automatically, and developers have to write manually some code to complete upgraded and newly added functionalities.

In cloud applications, the **business model** is tightly intertwined to the technical solution and the methodology defines several tasks in order to define the business model, taking as baseline the principles by Osterwalder [20]. Delivering a product is not the same as delivering a service in terms of organizational processes. Existing processes need to be analysed and redefined to adapt the organization to the new structure. Also, new processes related to the service delivery will have to be defined from scratch and to this end we have identified several processes that are affected by the shift of business models. These are: *Development Process, Updating Process, SLA Management, Helpdesk, Incidence Management, Marketing, Accountability, Cloud Provider and Roles Alignment.*

3.4. Post-Migration Phase. The quality of the migrated system needs to be verified, considering both behavioural (functional) and non-behavioural concerns such as performance or security. The migrated system has to operate similarly to the legacy system and needs to perform at least equally to the old system in terms of functionality and performance. The non-compliance of any of these requirements may cause the non-acceptance of the migrated system by the client. For this reason a validation and certification model is being used as part of the post-migration phase (Figure 3.5).

Once a feature or component has been migrated, its behavioural equivalence (across the legacy and migrated components) and the fulfilment of migration goals (i.e. non-functional requirements) need to be asserted. The first step of this validation covers the functional requirements of the system and is being performed against

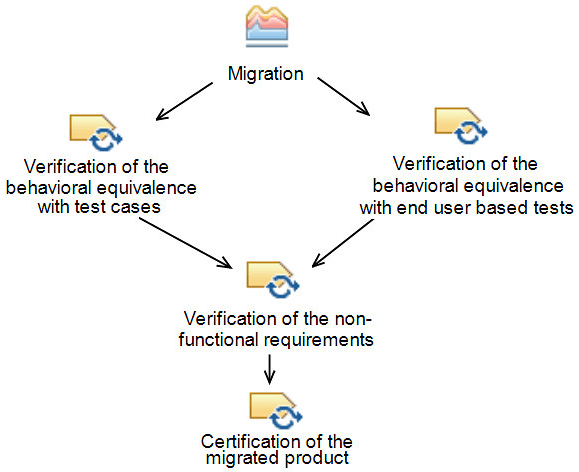

FIG. 3.5. *Post-migration EPF Model*

given test cases as well as provided by the end user test cases. The following step is the validation of the non-functional requirements, in terms of scalability, security, etc.

If the assessment fails, the process rolls back to the migration phase, in order to re-evaluate the migration requirements and the optimization/transformation strategies. Otherwise, the process moves to the next phase with concerns to the current feature. This phase iterates until all features have been migrated. If the assessment is successful, a certificate for the migrated project is being produced and following that the migrated artefact is being placed into the *ARTIST Repository* (explained in detail in the following section).

3.5. Migration Artefacts Reuse and Evolution Phase. The purpose of this supporting phase (Figure 3.6) is twofold: On the one hand, it provides tasks to foster reuse of migration artefacts inside and across migration projects, on the other hand it supports evolving the migrated system after a migration has been performed.

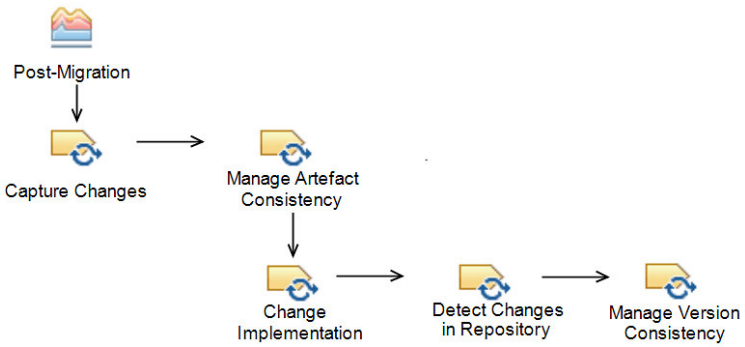

FIG. 3.6. *Artefacts Reuse and Evolution EPF Model*

During a migration project, a number of artefacts that are potentially reusable across projects are produced, such as meta-models, models, generic transformations and other information produced by the various framework tools. By reusing previous results, expensive operations such as big model extraction runs have to be performed only once. The artefacts are first checked for their reuse potential. Reusable artefacts are then published to the web-based *ARTIST Marketplace* if they should be publicly available or to the ARTIST Repository if they should be shared only within a controlled number of migration projects. The ARTIST Repository and Marketplace thus provide a central place to store, archive and organize MDE artefacts which could be merchandized as services in cloud marketplace environments [13].

The second purpose of this phase is to support maintenance activities during and after migration. This is achieved on the workspace level by supplying developers with information about the (modelling) artefacts in the workspace including their types, their relationships and dependencies and their history. This information is gathered from the workspace and from associated source code management and ticketing systems. One possible way to exploit this data is to provide traceability between artefacts and perform change notifications and change impact analyses.

On the artefact level, evolution is supported by a change description service. It can be triggered whenever new versions of one or more artefacts are produced. The service captures changes, detects inconsistencies in depending artefacts and supports the user in resolving these inconsistencies. Finally, the initial and the derived changes need to be implemented to complete the change workflow.

4. Architecture.

4.1. Architecture Overview. The architecture design is an integrated view of the functional blocks and components composing the ARTIST tooling, stereotyped with the main methodology block, i.e. pre-migration, migration and post-migration, to introduce the various components in turn.

The Migration Feasibility Assessment tooling comprises the Maturity Assessment Tool, the Business Feasibility Tool and the Technical Feasibility Tool. These tools interact with each other iteratively to estimate and report about the business and technical migration feasibility. This report is used by the Methodology Process Tool (described in next section) to tailor the particular migration process. The Migration Feasibility Assessment tools rely on some of the MDRE features provided by the Model Discovery and Understanding tools, offering low and high level abstraction platform-specific and independent models (PSM, PIM) of the legacy application.

The Cloud Metamodel and the model instances, describing target cloud providers and offerings, are provided by the Target Environment Specification tooling which comprises the Benchmarking, Performance Stereotype Classification and Profiling tools. These (meta)model artefacts are used during the modernization assessment (e.g. to specify cloud target and migration requirements) but also during the migration phase by the MDFE tooling (e.g. to specify requirements on the "cloudified" application and the cloud target environment). The MDFE Migration tooling comprises Target Specification, Optimization and Deployment tools, which supports the entire migration phase, such as the specification of requirements for the application and the target environment, target lookup and selection, application optimization (e.g. "cloudification") and application building and deployment.

The post-migration phase (Testing, Verification and Certification) is supported by a suite of Testing Tools and the SbSp (Service based Software providers) Certification Tool. Finally, the ARTIST Marketplace and ARTIST Repository realize the migration artefacts reuse and evolution phases and operate complementary to the other components so as to improve the effectiveness of various processes that these components perform.

4.2. ARTIST Suite. Following the design of the architecture, a concrete implementation approach for the ARTIST Suite, focusing on the support for the migration of both Java and .NET based applications is described in Figure 4.2.

The ARTIST suite comprises these main blocks of tools:

- **Browser-based (Web-based) ARTIST Suite**, targeting mostly end-users playing a **business role**. This tooling is intended to offer functionalities that require broader access to different types of users, compared to the more technical tooling which requires more specialized tools. Examples of tools in this block are: the Maturity Assessment Tool or the SbSp Certification Tool.
- **Eclipse RCP based ARTIST Suite**, targeting mostly end-users playing a **technical role**. Both the Eclipse platform itself and the Eclipse Modelling Project (EMP) tools, as baseline for ARTIST suite, are the natural choice looking to the selection of MDE techniques and OSS tools. Examples are those addressing technical functionalities, such as the Technical Feasibility Tool, the Model Discovery and Understanding, the Requirements Specification Tool, the Optimization Tool, the Target Generation Tool, the Deployment Tool or the different Testing Tools.
- **Sparx Enterprise Architect (EA)** [23], that includes a set of add-ons developed which support the discovery and target generation of .NET C# based applications.

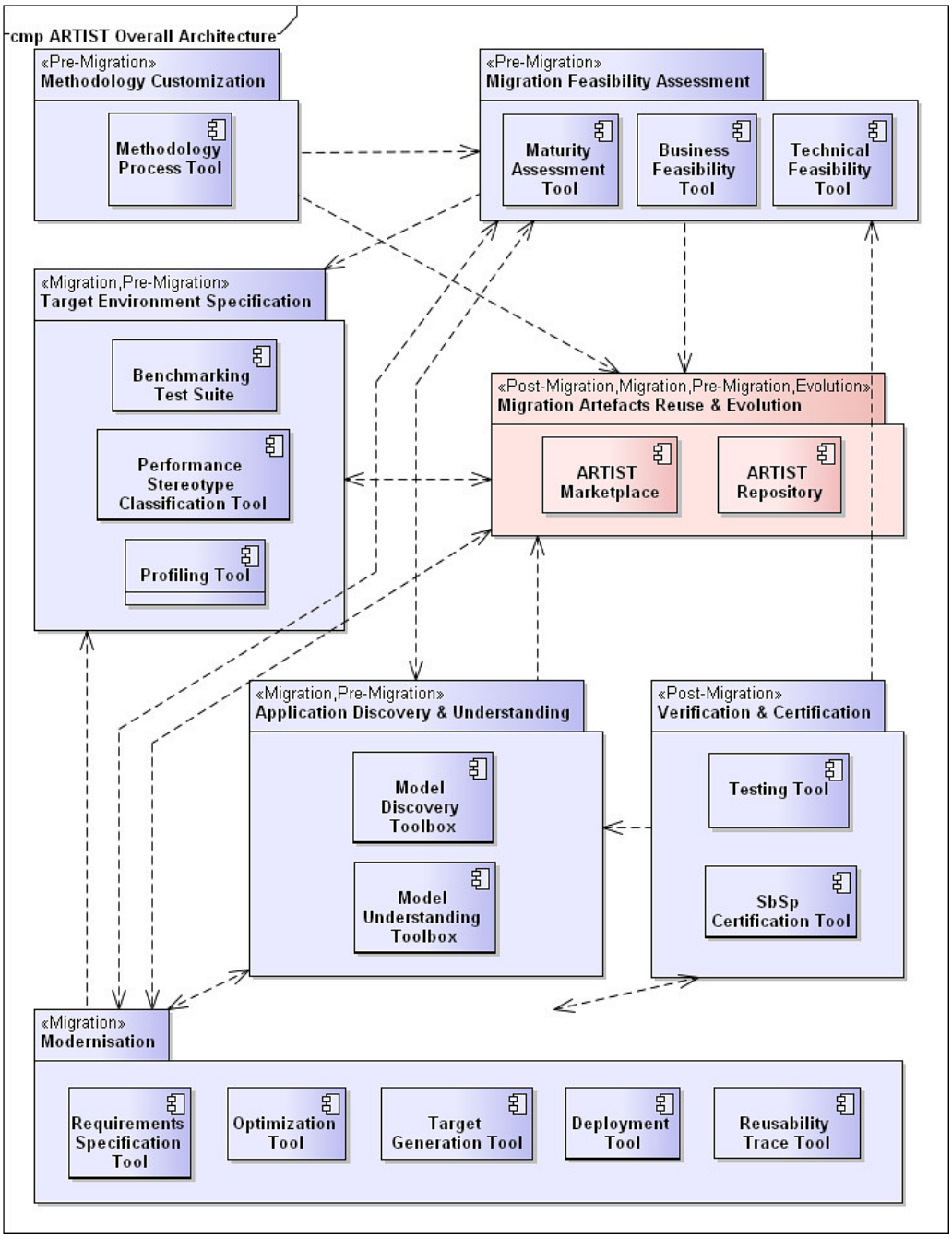

FIG. 4.1. *Overall Architecture Design*

The proposed suite supports the migration of both Java and .NET applications. However, Eclipse support for .NET development is somehow limited. .NET applications can be developed using Microsoft Visual Studio or other corresponding development environments. Our approach for the migration of .NET application relies on the usage of Sparx EA, during certain tasks of the methodology (notably during the Model Discovery and the Target Code Generation tasks). Thus, models of .NET applications will be obtained out of .NET projects using EA, and the final target code will be generated out of the migrated models using EA as well.

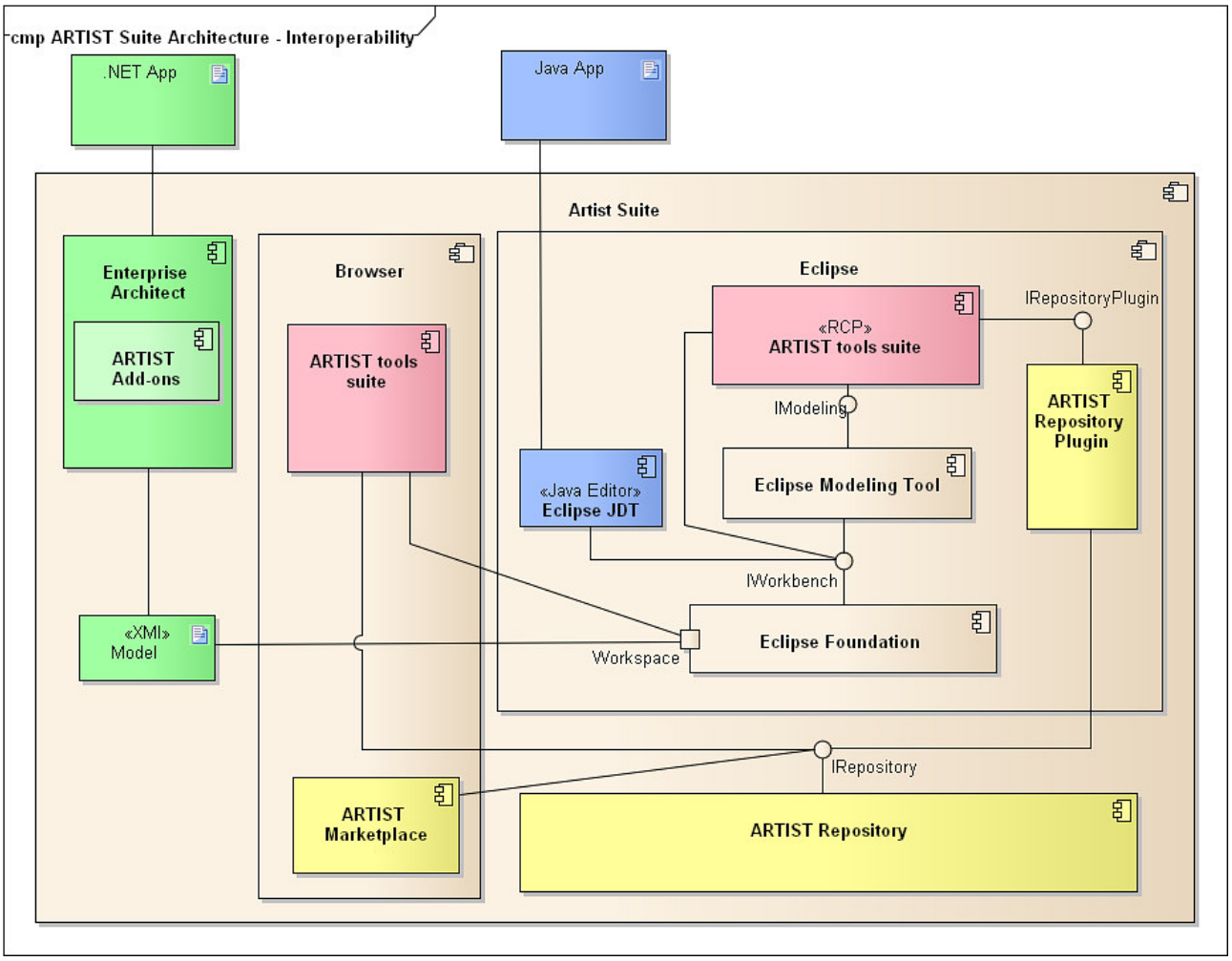

Fig. 4.2. *ARTIST Suite Architecture - Interoperability*

The remainder of the methodology tasks involving modelling are supported by Eclipse ARTIST tooling. The migrated .NET code can be compiled and assembled for deployment using Microsoft Visual Studio or another compatible development environment. Similarly, Java-based applications can be developed using the Eclipse JDT or another Java development environment. The projects of these Java applications can be then easily imported into the Eclipse workspace, ready to be managed by the tools. EA and Eclipse ARTIST tooling exchange models (notably UML ones) using the XML Metadata Interchange (XMI) [18] serialization format.

We foresee that the interactions between the browser-based and the Eclipse-based suites will be user driven and mediated through the Eclipse workspace and the repository. The user will save, locally on his/her workspace or in the repository, the artefacts produced during the performing of concrete tasks of the methodology and created using the browser-based tools. In turn, the user will import them within the Eclipse-based tools when required. The marketplace also offers browsing and searching functionalities to the end-users. Thus, the marketplace is the access point for users to the repository. Nevertheless, the various tools can access the repository directly through the repository plugin to retrieve/save artefacts provided that they have got the artefacts URIs.

Eclipse-based suite is tightly integrated within the Eclipse framework and the Eclipse Modelling Project (EMP) tools, using the Eclipse platform API and thus contributing to the Eclipse workbench. Artefacts produced and consumed by the Eclipse-based suite will be shared with other tool suite instances through the

repository by using the repository plugin. Alternatively, these artefacts can be stored locally in the Eclipse workspace for personal usage.

5. ARTIST Methodology Process Tool Implementation. The methodology is a modernization and migration solution that covers a wide range of application types, regardless of the underlying technologies, business models, operational modes, etc. In order to practically support this methodology, a core element was incorporated in the overall architecture: the Methodology Process Tool (MPT). This tool allows the customization and instantiation of the methodology based on the specific application requirements.

The Methodology Process Tool, exploiting the results processed and obtained during the modernization assessment, defines a customized modernization 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.

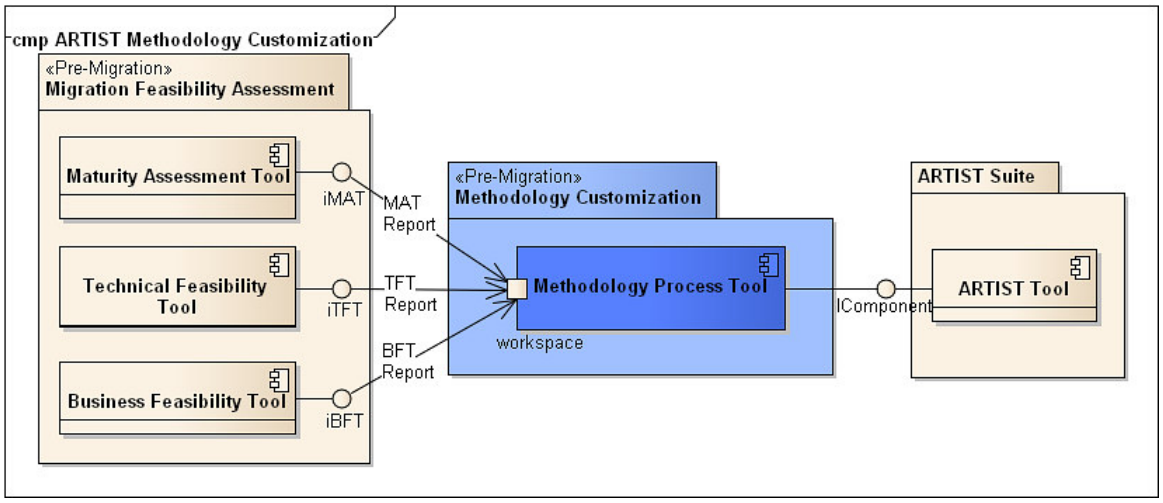

FIG. 5.1. *ARTIST Methodology Process Tool Dependencies*

MPT is the main component of the framework supporting the customization of the methodology at the end of the pre-migration phase. The methodology components are those which, based on the results of the modernization assessment (including the target specification), provide a tailored modernization blueprint. At the end of the pre-migration phase, after a positive early assessment (e.g. migration accepted), the remaining migration process (e.g. the rest of phases for completing the migration process) can be tailored to the specific characteristics of the migration project (i.e. legacy application and selected migration goals) using the Methodology Process Tool (MPT), which creates a specific blueprint for the migration project, that is, a specialization of the methodology process, rendered as a graphical process, showing tasks as visual elements (e.g. widgets) and related tasks for each phase within a group widget, logically connected through the methodology workflow. Moreover, each task widget includes links to the tools used to accomplish it, therefore selecting a task triggers the corresponding task tool in the integrated Tool Suite (e.g. Eclipse environment).

MPT uses reports generated during the early assessment phase, since the technical and business feasibility assessment reports, contains the required information to particularize the methodology to this concrete migration project. MPT can programmatically launch any required tool to accomplish any of the tasks described in the tailored methodology blueprint.

With respect to the overall architecture and methodology and the requirements posed by the objectives of the project, several approaches have been examined for the implementation of MPT. Currently we focus on a) a hybrid implementation for MPT, which combines features of a web-based platform and of the Eclipse environment and b) on an implementation using the SpikesTogether platform [24]. Both implementations are discussed in the following sections.

5.1. Hybrid (Web+Eclipse) Implementation. The MPT hybrid tool was built to take advantage of both the Web and the Eclipse based MPT architecture choices and combine them together to support the customization of the methodology from both any web browser (e.g. to support non-technical users, such as business analysts) or the Eclipse-based Tooling (e.g. to support technical users, such as modelers). In this way, the user is able to access MPT functionality either from an external Web browser, which connects to the web server that host the MPT, or within the Eclipse-based tooling, using the internal Eclipse Web Browser. Besides, this also allows the usage of Eclipse cheat sheets [5] to guide the user through the entire customized methodology and launch other Eclipse-based tools, as required by each task of that methodology.

Another challenge behind this implementation was to effectively exploit the representation of the methodology that was implemented using the Eclipse Process Framework, as described in the previous sections. For this reason, the methodology description outcome, which was in the form of simple static HTML pages, was transformed to XTML and integrated with Java Server Faces in order to achieve their dynamic customization according to the specific case under examination.

This implementation of the MPT provides to the end user the ability to upload and share existing reports in terms of maturity assessment, technical and business feasibility analysis, as produced from the corresponding tools, namely the MAT (Maturity Assessment Tool), TFT (Technical Feasibility Tool) and BFT (Business Feasibility Tool) tools respectively (see Figure 5.2).

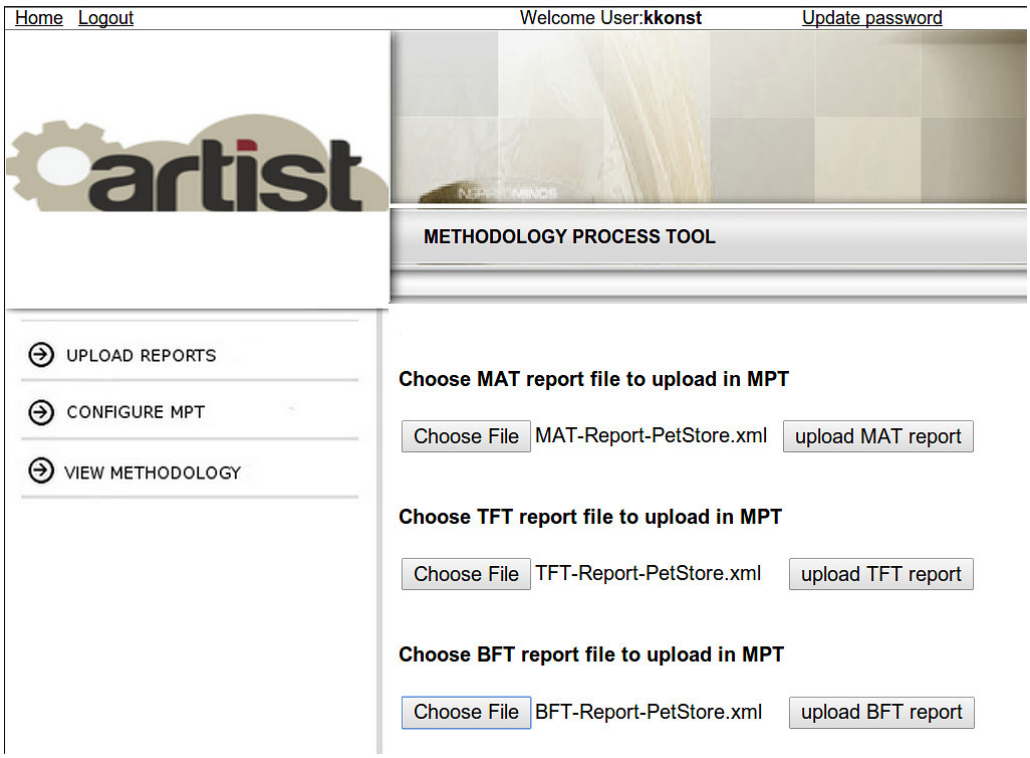

FIG. 5.2. *MPT Hybrid Tool: UPLOAD REPORTS*

Once the reports have been uploaded, the user is able to select among these reports and use the appropriate one to configure the methodology processes, according to the attached results for maturity, technical and business feasibility assessment. It should be noted that it is also possible to access existing reports that have been uploaded by the same user in the past for other projects as well as reports that other users are willing to share (see Figure 5.3).

By using the Methodology Configuration option, the selected report in XML format is being parsed and the extracted values are being stored and used against rules, the outcome of which may affect the content of

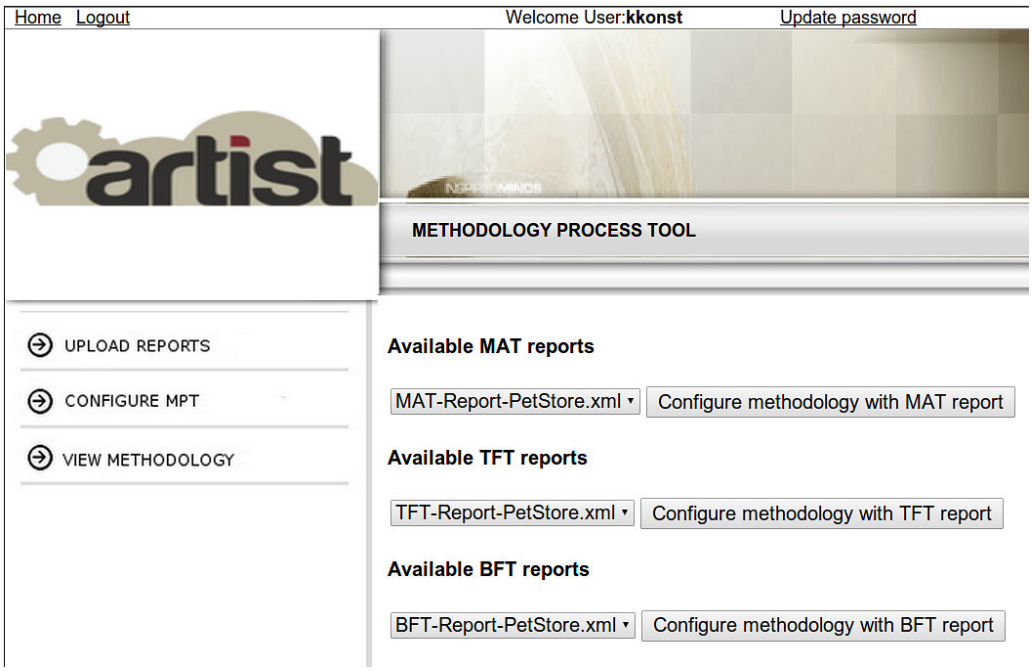

FIG. 5.3. *MPT Hybrid Tool: CONFIGURE MPT*

each methodology element (phase, task, activity) at different levels or even the flow that the user should follow in order to cloudify the project under examination.

Following the configuration of the methodology, the user can view the customized version of the EPF by clicking on the "View Methodology" button on the left pane. This action opens the index page of the EPF outcome in a new tab. An example for the methodology customization is described in the evaluation section.

5.2. SpikesTogether Implementation. A SpikesTogether-based approach has also been implemented focusing on accessibility towards business users. SpikesTogether [24] is developed on top of SharePoint [14], a well known web-based business collaboration platform. SpikesTogether is a tool which facilitates the development of line-of-business (LoB) solutions, extending the core functionality of SharePoint in particular key aspects.

SpikesTogether is built around two cornerstones: a) Adaptive Case Management and b) Collaborative Workflows. In the context of this work, application migration can be seen as a case to be managed while the migration process can be seen as a workflow. An actual application migration to the cloud typically involves many people which are supported by the collaborative aspect in the workflows. In addition a customized migration project is supported by the adaptive aspect in the case management.

Figure 5.4 shows a screenshot of a workflow in progress (i.e. the proposed methodology) on a particular item or migration project (i.e. SpikesTogether Migration). Tasks of the workflow in SpikesTogether are assigned to a user involved in a particular role. Every task has a certain outcome possibly leading to other tasks until the workflow (and consequently the application migration) has been completed. The figure also shows that the current migration is in the Migration phase which has not yet started while the pre-migration phase has already been completed. The example workflow used in this process can be seen in Figure 5.5. This workflow involves three different roles (i.e. Analyst, Modeler and Tester) each performing one particular step of the methodology process (i.e. pre-migration, migration and post-migration).

The SpikesTogether implementation guides the different users (in different roles) throughout the complete migration process and provides them with clear descriptions of the tasks that need to be performed using one or more of the tools in the tool suite. Identity/access management as well as security is being provided by the underlying SharePoint platform. Currently, the focus was around the step-by-step guidance of the user throughout the migration process while future work in this approach includes the customization of the

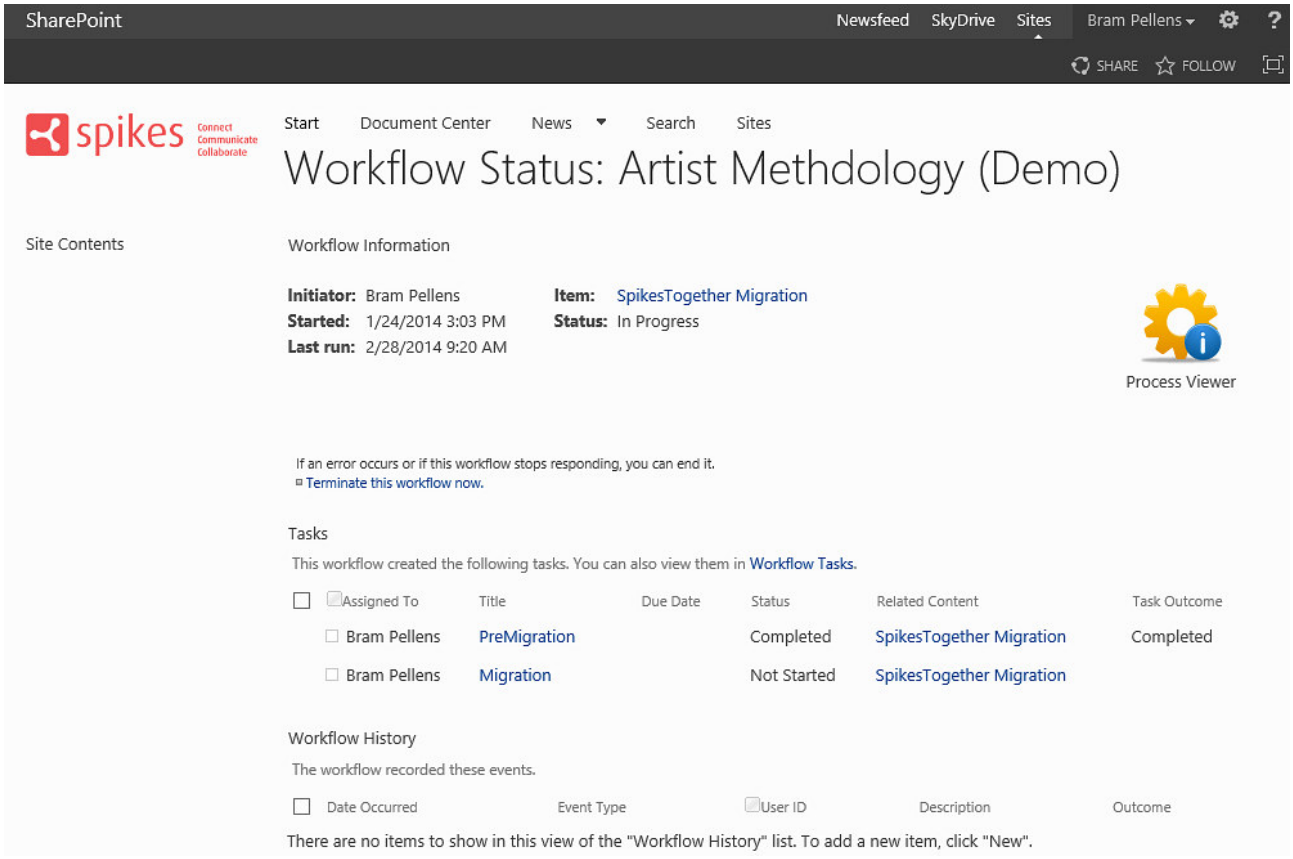

FIG. 5.4. *SpikesTogether MPT Implementation*

methodology (i.e. adaptation of the workflow).

6. Proof of Concept and Evaluation. In the previous sections, we described a generic but also detailed methodology for the migration of legacy software to modern target environments, which is also supported by a tool for its effective customization to the specific requirements of a particular migration project. In this section, the proof of concept for the Methodology Process Tool based on an experimental migration project is discussed. The proof of concept will show how specific steps of the generic methodology are customized for a concrete migration project based on the pre-migration analysis results. The Evaluation of the proposed methodology and its respective implementation, the Methodology Process Tool, was based in an experimental legacy J2EE Application, the PetStore [19]. In the following sections we describe the steps for the business and technical analysis of the application during the pre-migrations phase and then, based on the analysis results, we describe the rule-based customization of the methodology through MPT.

6.1. PetStore Business Case. Our objective is to tackle the problem of software modernization under two perspectives, the technical one and the business one. This is the reason why **we need to define the business case of the PetStore in order to evaluate and test all the tools under both perspectives**:

SME-Software is an English company producing and marketing software products targeting SME Enterprises. Their customers include Hotels, Restaurants, Stores, Chartered Accounting Firms, Employment Law specialists. PetStore application started as a management system or ERP for stores selling pets, and evolved as a web portal for selling pets over Internet.

Nowadays SME-Software sells its application by licensing (to be renewed each year) and offers the possibility to customize it through consultancy services via ad-hoc projects. This customization includes the purchase of the new infrastructure (Application server, database server, etc.), installation of the adequate application server

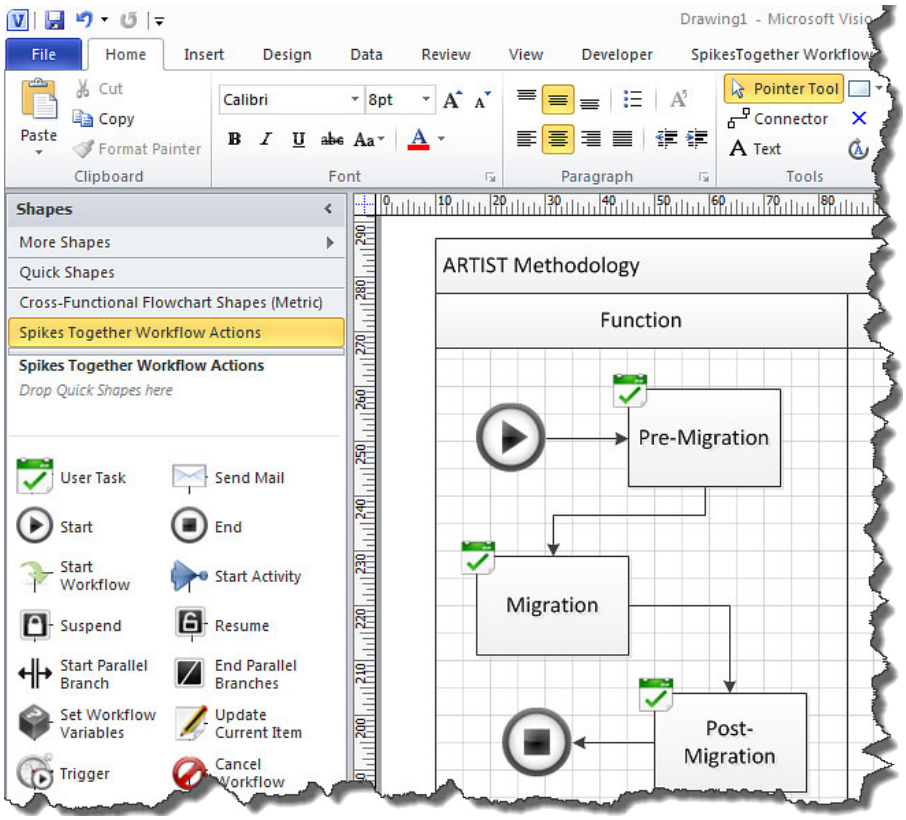

FIG. 5.5. *SpikesTogether MPT Usage Example*

and database server software versions to fit the PetStore requirements, and customization of the interface layer. The customers may contract these services to companies other than SME-Software. Optionally, customers can contract maintenance support.

With the emergence of cloud computing and the SaaS business model, SME-Software managers have start thinking about offering their Software as a Service. Many customers have shown willingness to contract the service instead of buying the product and the infrastructure required to install it.

SME-Software's CIO is aware of the emergence of cloud computing and he has participated in discussions where the advantages of cloud computing have been exposed. He notices that their customers are demanding changes in the way they sell and offer their product but he is still reluctant to change because the following aspects are not known:

- *implications to change from SaaP (Software as a Product) to SaaS*
- *time and costs implied from a potential migration*
- *which IaaS provider is the best, and which implications this could have*
- *which risks (market and technical) are associated to this change*
- *which business processes are affected and how to adapt them to the new selling model.*

6.2. PetStore Software and Business Case Assessment. MPT tool will personalize the methodology for each migration project relying on the results obtained in the pre-migration phase by the different tools (MAT, TFT and BFT). From MAT, the migration goals and high level recommendations will be used, from TFT, the list of migration tasks and affected components and finally, from BFT the selected business scenario. By combining all this information, MPT will personalize the generic methodology for each specific migration project.

6.2.1. MAT results. The Maturity Assessment Tool results on application positioning with respect to its cloud maturity are the following:

- **Migration Goals**
  - Migrated programming language: J2EE
  - Multitenancy level: Multitenant
  - Database scalability: High scalability of data
  - Interoperability requirements: Yes, ERP
  - SLA requirements: System availability, performance and QoS, Outages covered, Data integrity, business continuity
  - Configuration requirements: User personalization
  - Authorization requirements: ID, Password
  - Cloud provider: Amazon, Public Cloud
  - Elasticity: Component / self-made means to ensure its elasticity
- **High level recommendations**
  - Model the application to have a better knowledge of the application
  - Redesign the database to be multitenant
  - Redesign the architecture and distribution of stateful/stateless nodes
  - Integrate a module of concurrent users monitorization
  - Define authentication mechanisms
  - The coexistence of the two business models may bring difficulties in the support processes
  - Define new SLAs taking into account the infrastructure part

6.2.2. TFT results. The Technical Feasibility Tool is aiming at estimating the efforts required to migrate a legacy application to a selected target cloud environment, fulfilling some migration goals and requirements. This effort estimation is broken down into each legacy component and the migration tasks required migrating it. The effort required to accomplish each migration task is derived from the complexity of the task and the complexity inherent to the legacy code (component complexity). TFT analysis results are summarized in the following table. In the example shown, the different components of the PetStore have been identified ("component" column) and the following metrics have been obtained for each of them:

- **Component complexity**: This is the complexity related to the legacy software. In this proof of concept, just for the sake of simplicity, component complexity is just measured by the number of files comprising the component: 1 XML file for J2EE resources (complexity 1.0) and 40 Java classes for PetStore (complexity 40.0), 1 SQL file for PetStore data schema, 1 bundle file (installation) for J2EE Server and Non-SQL Server. More elaborated complexity based on the Maintainability metric and SW/OOP/Coupling metrics will be computed by TFT.
- **Task**: This is the migration task suggested by TFT (following a rule based approach) to migrate the corresponding component.
- **Task Type**: Each task is been characterized by its type. Task taxonomy (including Installation and configuration tasks, Data sources related tasks, Code refactoring tasks, Connection/Configuration tasks) has been defined in the context of the TFT where the knowledge representation of expert judgement with respect to the complexity of the task is embedded.
- **Task Complexity**: The task complexity is obtained by the TFT based on the selected task complexity level and expert heuristics and/or historical data.
- **Task Effort**: This is the required effort to complete the task. The effort is computed based on the component complexity, task complexity and expert heuristics and/or historical data.

6.2.3. BFT results. In the business feasibility analysis the following business scenarios were identified:

- **PetStore In-house**: This scenario represents the Business Model as-is for SME Software Co. at the time of the assessment. PetStores buy licenses from SME Software Co. PetStores rely on external organizations for maintenance of hardware and software and provisioning of resources (e.g. hardware, energy). Customers buy pets from PetStores using the PetStore Software.
- **PetStore on IaaS**: This scenario presents the Business Model assuming that the PetStores rely on IaaS Providers to operate the PetStore service: PetStores purchase SW License from SME Software Co.

TABLE 6.1
TFT Results

Component		Task			
Name	Compl.	Name	Type	Compl.	Effort
J2EE Server	1.0	App Server Installation & Configuration	Installation and Configuration	2.0	2.0
Non-SQL Server	1.0	Non-SQL persistence framework installation and configuration	Installation and Configuration	2.0	2.0
PetStore Web App	40.0	PetStore Persistence Layer re-coding based for Non-SQL persistence framework	Code Refactoring	5.0	40.0
PetStore Web App	1.0	PetStore data schema refactoring for Non-SQL persistence framework	Data Source	5.0	8.0
Non-SQL Server	1.0	PetStore data dump into Non-SQL persistence framework	Data Source	1.5	1.0
JDBC Resource	1.0	PetStore JDBC Resource reconfiguration	Connection-Configuration	1.0	0.1
Connection Pool	1.0	PetStore connection pool reconfiguration	Connection-Configuration	1.0	0.1
				Total	53.2

In this scenario, the SW is licensed at a lower price. The PetStores rely on IaaS Providers to operate the PetStore service. In turn IaaS purchase resources (e.g. hardware, energy) from external resource providers. Customer buys pets from PetStores using the PetStore Software. PetStores pay royalties to SME Software Co. for each transaction completed by means of the PetStore services

- **PetStore on PaaS without subscription fee**: SME Software integrates the PetStore software relying on the platform of a PaaS Provider. SME Software pays the PaaS provider to operate the PetStore service. The IaaS buy resources (e.g. hardware, power) from external resource providers. Customer buys pets from PetStores using the PetStore Software. PetStores pay royalties to SME Software Co. for each transaction completed by means of the PetStore services.
- **PetStore as Cloud Provider**: PetStore pays a subscription to the PetStore service operated directly by SME Software Co. using in-house infrastructure. SME Software Co. pays for both the maintenance the infrastructure and resources acquisition. PetStore pay royalties to SME Software Co. for each transaction completed by means of the PetStore services.

6.3. Methodology Customization and Instantiation for Proof of Concept. The MPT is developed with a set of rules for the customization of the methodology based of the parameters of each migration scenario. In the proof of concept, we present the following example rules which were applied to the pre-migration analysis results of the PetStore application. These example rules are based on the technical and business migration goals defined by the user and extracted from MAT.

Example Rule #1 (technical level): This rule examines a) the parameter *databasereq* which refers to the database requirements (requirements elicitated by the application owner for the database) and b) the

parameter *targetplat* which refers to the (cloud) target platform selected by the application owner (where the application will be deployed). This rule is triggered with the value *high_escalability* for *databasereq* parameter (high scalability of database is required) and with the value *AEC2* for *targetplat* paremeter (Amazon EC2 Infrastructure).

Listing 1
Example Rule for Customization on Technical Level

```
If (databasereq == high_escalability AND targetplat == AEC2)
 1. EMREQ task has to be personalized as follows:
  a. Annotate PIM with AWS requirements;
  b. Annotate the backend component with the requirement
     stereotype "highly scalable database";
 2. OPTAPPFEA task has to be personalized as follows:
  a. Data schema refactoring and optimization to
     target NoSQL persistence frameworks;
  b. Data dump transformation to fit the data
     into the target NoSQL persistence frameworks;
  c. Persistence layer adaptation based on the
     selected target NoSQL persistence frameworks;
end If
```

Example Rule #2 (business level): This rule examines the parameter *bil_rule* which refers to the billing rule that the application owner intends to apply once the application is migrated. This rule is triggered with the value *use* for this parameter that implies direct use of the application.

Listing 2
Example Rule for Customization on Business Level

```
If (bil_rule == use)
 1. REVE task has to be personalized as follows:
  a. Service price definition based on use;
  b. Monitoring and billing component to be incorporated:
     Link to EMREQ and OPTAPPFEA technical tasks;
 2. EMREQ task has to be personalized as follows:
  a. Annotate PSM with monitoring and billing component;
 3. OPTAPPFEA task has to be personalized as follows:
  a. Create the monitoring component at PSM level;
end If
```

Based on the pre-migration analysis and the rules, the methodology is customized so as to define the exact steps that should be followed to accomplish the EMREQ (Express Migration Requirements) task for the specific migration project. Figures 6.1 and 6.2, show the output of two different customizations that affect the way the migration requirements are expressed, covered by the EMREQ task.

Figure 6.1 shows the default view of the page that connects to the EMREQ task. This version appears when the user has not customized the methodology following the process described earlier, or when the reports used for the customization do not trigger a rule that affects the specific task. On the other hand, Figure 6.2 shows a customized version of the same web page in which a rule triggered the addition of extra steps that inform the user that the platform independent model and the backend component should be annotated with specific requirements. In a similar way, all methodology tasks can be customized to the requirements of the particular project so as to ease the navigation through the various tasks and activities simplifying considerably the migration workflow.

7. Conclusions. In this paper we presented a novel software migration approach, which is capable to empower the technical and business capabilities of legacy applications by its modernization and migration to modern cloud environments. The ARTIST Migration Methodology and Framework, as an "all-in-one" solution allows the legacy applications to exploit the offerings of the cloud providers, enrich their unique characteristics and benefit from offering them as services in a potentially global market.

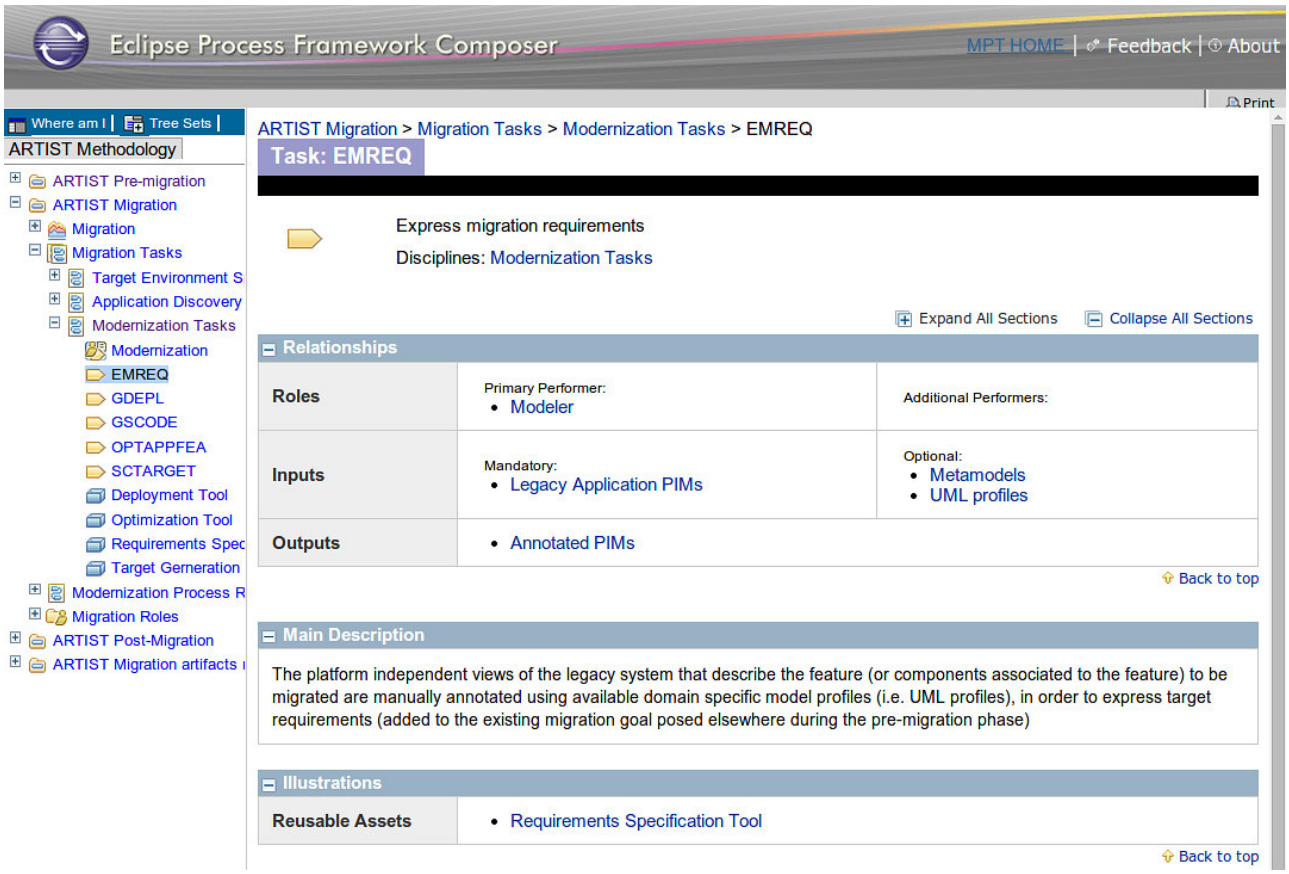

FIG. 6.1. *MPT hybrid tool: Default view of the EMREQ task page (no active rule)*

The added value of the proposed methodology can be summarized as follows: 1) it includes a feasibility analysis before any investment is actually made, 2) it is focused on cloud-compliant architectural issues at both application and infrastructure levels, 3) it includes business model issues that are strongly linked to the technical decisions that are made, 4) it takes into account the impact of the business model shift on the organization processes, 5) it fosters re-usability and automation, 6) it globally prepares the software for its evolution.

The methodology, covering both the technical and business aspects of the migration process, is tailored to the needs of the specific application and guides the owners and developers in every step so as to take full advantage of their software in an effective and productive manner. In addition, a complete set of tools is supporting each methodology task realizing modern analysis and model-driven engineering approaches, allowing when possible the reuse of artefacts. The core of this rich software suite is the Methodology Process Tool which is in charge of the customization and instantiation of the generic methodology based on the pre-migration analysis results of each migration project and the customization rules. The Methodology Process Tool is currently available in two implementations so as to integrate smoothly to the development workflow and preferences of both the JAVA and .NET development communities while further extensions are foreseen to further improve its capabilities and effectiveness.

Acknowledgments. 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.

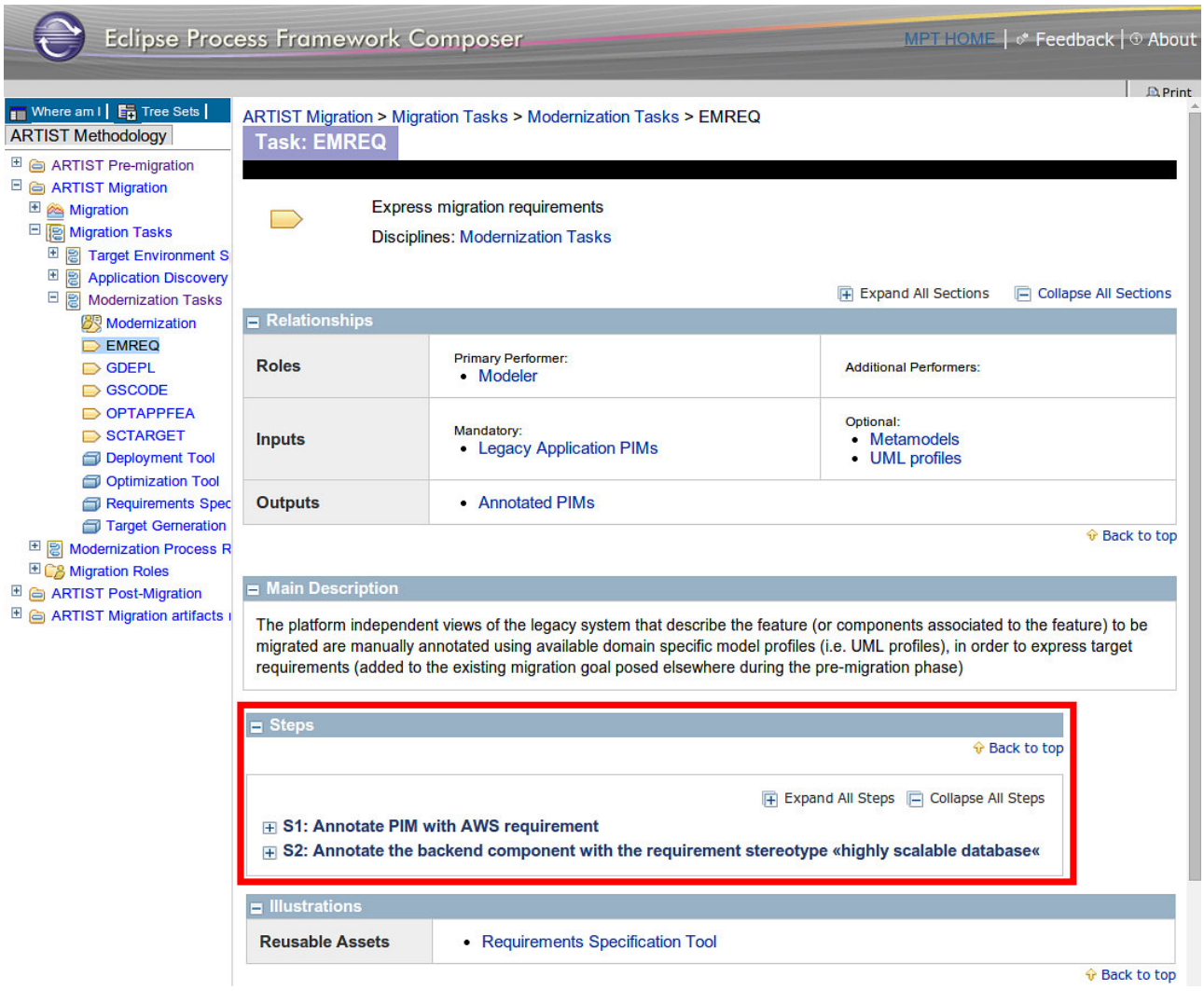

FIG. 6.2. *MPT hybrid tool: Customized view of the EMREQ task page (technical level rule active)*

REFERENCES

[1] ARTIST PROJECT. Available online at: `http://www.artist-project.eu`.

[2] A. BAGNATO, M. SERINA, AND M. MARCO, *In the MOMOCS Tool Suite: a XIRUP Transformation Tool to achieve complex system modernization*, Proceedings OF Model-Driven Modernization OF Software Systems 2008, (2008), p. 101.

[3] H. BRUNELIERE, J. CABOT, F. JOUAULT, AND F. MADIOT, *Modisco: a generic and extensible framework for model driven reverse engineering*, in Proceedings of the IEEE/ACM international conference on Automated software engineering, ACM, 2010, pp. 173–174.

[4] R. BUYYA, C. S. YEO, S. VENUGOPAL, J. BROBERG, AND I. BRANDIC, *Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility*, Future Generation computer systems, 25 (2009), pp. 599–616.

[5] ECLIPSE FOUNDATION, *Eclipse Cheat Sheets*. Available online at: `http://www.eclipse.org/pde/pde-ui/articles/cheat_sheet_dev_workflow`.

[6] ECLIPSE FOUNDATION, *Eclipse Modeling Project*. Available online at: `http://www.eclipse.org/modeling`.

[7] ECLIPSE FOUNDATION, *Enterprise Process Framework - EPF*. Available online at: `http://www.eclipse.org/epf`.

[8] G. KOUSIOURIS, G. GIAMMATTEO, A. EVANGELINOU, N. GALANTE, E. KEVANI, C. STAMPOLTAS, A. MENYCHTAS, A. KOPANELI, K. RAMASAMY BALRAJ, D. KYRIAZIS, T. VARVARIGOU, P. STUER, AND L. ORUE-ECHEVARRIA, *A multi-cloud framework for measuring and describing performance aspects of cloud services across different application types*, in 4th International Conference on Cloud Computing and Services Science (CLOSER 2014), 2014.

[9] G. KOUSIOURIS, D. KYRIAZIS, A. MENYCHTAS, AND T. VARVARIGOU, *Legacy applications on the cloud: Challenges and enablers*

focusing on application performance analysis and providers characteristics, in Cloud Computing and Intelligent Systems (CCIS), 2012 IEEE 2nd International Conference on, vol. 2, IEEE, 2012, pp. 603–608.

[10] G. Lewis, E. Morris, and D. Smith, *Service-oriented migration and reuse technique (smart)*, in Software Technology and Engineering Practice, 2005. 13th IEEE International Workshop on, IEEE, 2005, pp. 222–229.

[11] P. Mell and T. Grance, *The NIST definition of cloud computing*, National Institute of Standards and Technology, 53 (2009), p. 50.

[12] A. Menychtas, C. Santzaridou, G. Kousiouris, T. Varvarigou, L. Orue-Echevarria, J. Alonso, J. Gorronogoitia, H. Brunelière, O. Strauss, T. Senkova, et al., *ARTIST Methodology and Framework: A novel approach for the migration of legacy software on the Cloud*, in 2nd Workshop on Management of resources and services In Cloud And Sky computing (MICAS 2013), 2013.

[13] A. Menychtas, J. Vogel, A. Giessmann, A. Gatzioura, S. G. Gomez, V. Moulos, F. Junker, M. Mller, D. Kyriazis, K. Stanoevska-Slabeva, and T. Varvarigou, *4caast marketplace: An advanced business environment for trading cloud services*, Future Generation Computer Systems, (2014)

[14] Microsoft, *Microsoft SharePoint*. Available online at: `http://office.microsoft.com/en-us/sharepoint`.

[15] P. Mohagheghi and T. Sæther, *Software engineering challenges for migration to the service cloud paradigm: Ongoing work in the remics project*, in Services (SERVICES), 2011 IEEE World Congress on, IEEE, 2011, pp. 507–514.

[16] Open Management Group, *Architecture-Driven Modernization (ADM)*. Available online at: `http://adm.omg.org`.

[17] Open Management Group, *Software & Systems Process Engineering Metamodel Specification (SPEM) Version 2.0*. Available online at: `http://www.omg.org/spec/SPEM/2.0`.

[18] Open Management Group, *XML Metadata Interchange - XMI*. Available online at: `http://www.omg.org/spec/XMI`.

[19] Oracle, *Java Pet Store J2EE Experimental Application*. Available online at: `http://www.oracle.com/technetwork/articles/javaee/petstore-137013.html`.

[20] A. Osterwalder and Y. Pigneur, *Business model generation: a handbook for visionaries, game changers, and challengers*, John Wiley & Sons, 2010.

[21] S. Rugaber and K. Stirewalt, *Model-driven reverse engineering*, Software, IEEE, 21 (2004), pp. 45–53.

[22] D. C. Schmidt, *Model-driven engineering*, COMPUTER-IEEE COMPUTER SOCIETY-, 39 (2006), p. 25.

[23] Sparx Systems, *Enterprise Architect*. Available online at: `http://www.sparxsystems.com/products/ea`.

[24] Spikes, *Spikestogether*. Available online at: `http://www.spikestogether.com`.

[25] I. Warren and J. Ransom, *Renaissance: a method to support software system evolution*, in Computer Software and Applications Conference, 2002. COMPSAC 2002. Proceedings. 26th Annual International, IEEE, 2002, pp. 415–420.

[26] B. Wu, D. Lawless, J. Bisbal, J. Grimson, V. Wade, D. O'Sullivan, and R. Richardson, *Legacy systems migration-a method and its tool-kit framework*, in Software Engineering Conference, 1997. Asia Pacific... and International Computer Science Conference 1997. APSEC'97 and ICSC'97. Proceedings, IEEE, 1997, pp. 312–320.

Edited by: Florin Fortiş
Received: Mar 1, 2014
Accepted: Jun 24, 2014