ARTIST

FP7-317859

Advanced software-based seRvice provisioning and migraTIon of legacy Software

Deliverable D5.3.2

Technical Feasibility Tools

Editor(s):	Burak Karaboğa, Jesus Gorroñogoitia
Responsible Partner:	Atos
Status-Version:	Final – v1.0
Date:	15/09/2014
Distribution level (CO, PU):	PU

Project Number:	FP7-317859
Project Title:	ARTIST

Title of Deliverable:	D5.3.2 Technical Feasibility Tools
Due Date of Delivery to the EC:	30/09/2014

Workpackage responsible for the Deliverable:	WP5 – Modernization Assessment
Editor(s):	Atos - Burak Karaboğa, Jesus Gorroñogoitia
Contributor(s):	Atos – Burak Karaboğa, Jesus Gorroñogoitia Tecnalia – Zurik Corera, Juncal Alonso
Reviewer(s):	TUWien – Manuel Wimmer
Approved by:	All Partners
Recommended/mandatory readers:	WP6, WP7, WP8, WP9, WP11

Abstract:	These tools analyse the complexity and coupling of the source code of the legacy product, its data schema and database technologies, and create high level abstract models out of the legacy source code that address the concerns related to its migration into the desired target cloud platform, providing an estimation of the work (effort) that would be needed to transform the legacy product into that target platform.
Keyword List:	Technical feasibility, Complexity, Maintainability
Licensing information:	TFT is delivered under EPL license. The document itself is delivered as a description for the European Commission about the released software, so it is not public.

Document Description

Document Revision History

Version	*Date*	*Modifications Introduced*	
		Modification Reason	*Modified by*
v0.1	01/09/14	First contributions for M24 release	ATOS
v0.2	04/09/14	Initial contributions to SCC section	TECNALIA
v0.3	10/09/14	Final contributions to SCC section	TECNALIA
V0.4	12/09/14	Final contributions and updates	ATOS
V1.0	15/09/2014	Final reviewed version	ATOS

Table of Contents

Table of Figures

Terms and abbreviations

EC	European Commission
TFT	Technical Feasibility Tool
MAT	Maturity Assessment Tool
EMF	Eclipse Modelling Framework
UML	Unified Modelling Language
JAXB	Java Architecture for XML Binding
URL	Uniform Resource Locator
MPT	Methodology Process Tool
MUT	Model Understanding Tool

Executive Summary

This document describes the Technical Feasibility Tool (TFT), which studies the feasibility of the legacy application's migration to cloud within the context of ARTIST pre-migration approach, along with detailing the Software Complexity Component (SCC) which plays a major role in this feasibility analysis.

The main motivation behind TFT is to assist the user to have a better understanding of the migration to cloud process and the efforts required to accomplish this process in the ARTIST pre-migration phase.

The purpose of this deliverable is to describe the modifications and new features developed and released in the second version of the Technical Feasibility Tool detailing its design and implementation and thus give an idea about the tool's usability, capabilities and potential.

M24 version of the Technical Feasibility Tool introduces substantial amount of improvements and changes to the existing UI elements and backend components. Besides these updates, the tool's functionalities and capabilities are also extended with new visual elements and backend components.

Regarding the visual improvements; Migration Efforts View has been implemented to show the user estimated efforts for each component and migration strategy which also contributed to Inventory View to be more user friendly by reducing the information shown on it. Inventory View has been improved with support for nested components, report creation and a new dialog which assists the user on changing the assigned migration strategies on components. Finally Migration Goals View has been updated to reflect the selection changes of target platform immediately on the inventory view with updated migration strategy suggestions.

Regarding the implementation improvements; TFT and SCC has been integrated in order to make use of the metrics information collected by SCC in effort estimation computations for migration strategies and model components, suggestion updates upon user modification of migration suggestions improved, migration strategy taxonomy has been defined and incorporated into Drools, migration strategies (xml file) is restructured to support multiple target platforms per strategy and serialized to be easily shareable with MPT in the context of WP6, component models improved to include dependency relationship among components and support for nested component structure and finally the rule knowledge base has been extended to host more than 100 new rules supporting migration strategies for target platforms GAE, AWS and Azure. In this second year the SCC has been enhanced with the calculation of metrics analysing the relationship among classes (such as aggregation or generalization) as well as the calculation of metrics at component level. As a result, the current prototype provides an initial approximation of the maintainability metric (some metrics are still missing and will be calculated for the last version of the prototype). The support to the C# solution has also been improved for this version. The current prototype supports the calculation of the metrics also for C# code, at the same level as it does for Java based code.

The document presents the missions, scopes, functional descriptions, technical approaches, download & installation instructions and user manuals of these components comprising the current version of the Technical Feasibility tools at M24. The deliverable focuses on improvements and extensions upon the initial version of TFT provided on M12 and it refers to deliverable D5.3.1 for unchanged features and details of the tool.

The next iteration of this deliverable is planned to be at M30 with increased capabilities and stability, a better integration and communication between TFT components and external dependencies of TFT (e.g. other ARTIST tools). The next version is planned to extend the existing knowledge base with new rules to handle more complex scenarios, make smarter migration strategy suggestions, estimate efforts needed for migrations more precisely, be more user friendly and responsive.

1 Introduction

1.1 About this deliverable

The purpose of Technical Feasibility Tools (TFT) is to offer an early technical analysis of the migration of an existing non-Cloud compliant application, by conducting an analysis of the Maturity Assessment Tool's (MAT) report, and the high level component models of the application extracted by the Model Understanding Toolbox, in order to obtain a set of component maintainability metrics, migration strategy suggestions and migration effort estimations, which altogether offers a migration feasibility technical report. In the context of ARTIST, TFT assists the user in the pre-migration phase by displaying the analysis results and suggesting migration strategies to the users about their existing systems in a responsive way.

TFT is comprised of different components which have different roles in the analysis process of the migration candidate legacy application. These components are; component detector, strategy suggestor, software complexity component, effort estimator and TFT repository, last of which is not offered in this version. Component detector is responsible for analyzing the high level component model obtained from MUT and population of the component data model in the memory. Strategy suggestor examines this data model and migration goals obtained from MAT in order to suggest appropriate migration strategies for each component. Software complexity component (SCC) is responsible for the analysis of the source code, class model and the component model to calculate the complexity metrics for the whole application and each individual high level component. The effort estimator uses the complexity values detected by SCC and the migration strategies suggested by strategy suggestor to estimate the effort needed to migrate each component and also the whole application to the target platform.

This deliverable describes the technical and functional aspects, the role in the overall ARTIST solution, installation details and user manuals of TFT components in detail while also focusing on the new features and changes that has been implemented on them for the M24 Prototype release.

1.2 Document structure

This document is divided into three main sections describing TFT in general, detailing the SCC component of TFT and lastly presenting the interaction between TFT components. The document is concluded followed by the References and the Appendix.

This version of the deliverable focuses on the updates and new features with respect to the previous release, therefore the unchanged sections/subsections will refer to the old version. The first two sections are divided into subsections; Implementation and Delivery and Usage. Implementation sections focus on functional and technical aspects of the components in several subsections of their own. These subsections include information about functional aspects, fitting into overall ARTIST solution and technical aspects such as prototype architecture, components description and technical specifications. Delivery and Usage section focuses on what is delivered, how to download, install and use this deliverable. The package information subsection describes the package contents; the classes, resources and library files used by the component. Download and Installation Instructions sections explain how to download the prototype from the repository, how to install it as well as the prerequisites in detail. The User Manual section describes the Eclipse Views and other elements that the user can interact with that are offered by each component of TFT.

Interaction of TFT Components section focuses on the communication between TFT IDE and SCC components, describing the API offered by SCC and how TFT uses this API to calculate effort estimations for suggested migration strategies.

The Appendix section describes the taxonomy of migration tasks and lists the remarkable migration strategies offered by TFT in two subsections.

2 TFT

2.1 Implementation

2.1.1 Functional Description

The Technical Feasibility Tool (TFT), whose technical description is provided in [1], aims at supporting ARTIST users on the early technical assessment of the migration of a legacy application to the cloud. At this early stage (e.g. pre-migration phase in ARTIST Methodology), the ARTIST users need to be supported to evaluate the feasibility of the migration, attending its technical aspects, since even for a very simple legacy application, its migration to the cloud could be a complex process, that may require non negligible efforts and concrete expertise to be accomplished. Moreover, the support for decision making at this early pre-migration stage requires a detailed breakdown of the migration process into a set of technical tasks, not only to estimate their required efforts, but also 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.

In this context, the role of this version of TFT prototype is to offer technical information about the legacy application itself, suggest technical strategies to migrate its components to the cloud and to offer migration effort estimations for each component marked for migration by analyzing the information in the migration goals of the MAT report, the high level view of the application provided by Model Understanding Tool (MUT), SCC's metric analysis as well as user preferences [3] and lastly generation of the TFT report for the consumption of Methodology Process Tool (MPT).

The prototype is responsive to user interactions such as, modifications on MUT's output, elimination of migration goals from MAT's report and re-assignment of migration strategy suggestions on components. These interactions trigger the analysis process of TFT and thus updating the migration strategy suggestions, complexity and effort estimations.

TFT may be considered as a high-level technical analysis for the whole legacy application. What MUT provides to TFT is a filtered, refined model of the legacy system and this output requires a certain analysis of structure to be created as well but this reveals what the system "is". TFT, on the other hand, aims to find out what the application "can" be or "should" be. This is why TFT's analysis is considered to be high-level.

2.1.1.1 Fitting into overall ARTIST solution

Next picture depicts the main TFT components and their dependencies and/or relationships with other ARTIST components. Note that not all TFT foreseen dependencies and interactions, as described in next figure, are implemented in M24 prototype. As commented in this section, current version of TFT integrates MAT reports and MUT high level models. For further details see [5] D5.3.1 Technical Feasibility Tools and [6] D6.4.1 ARTIST Integrated Architecture.

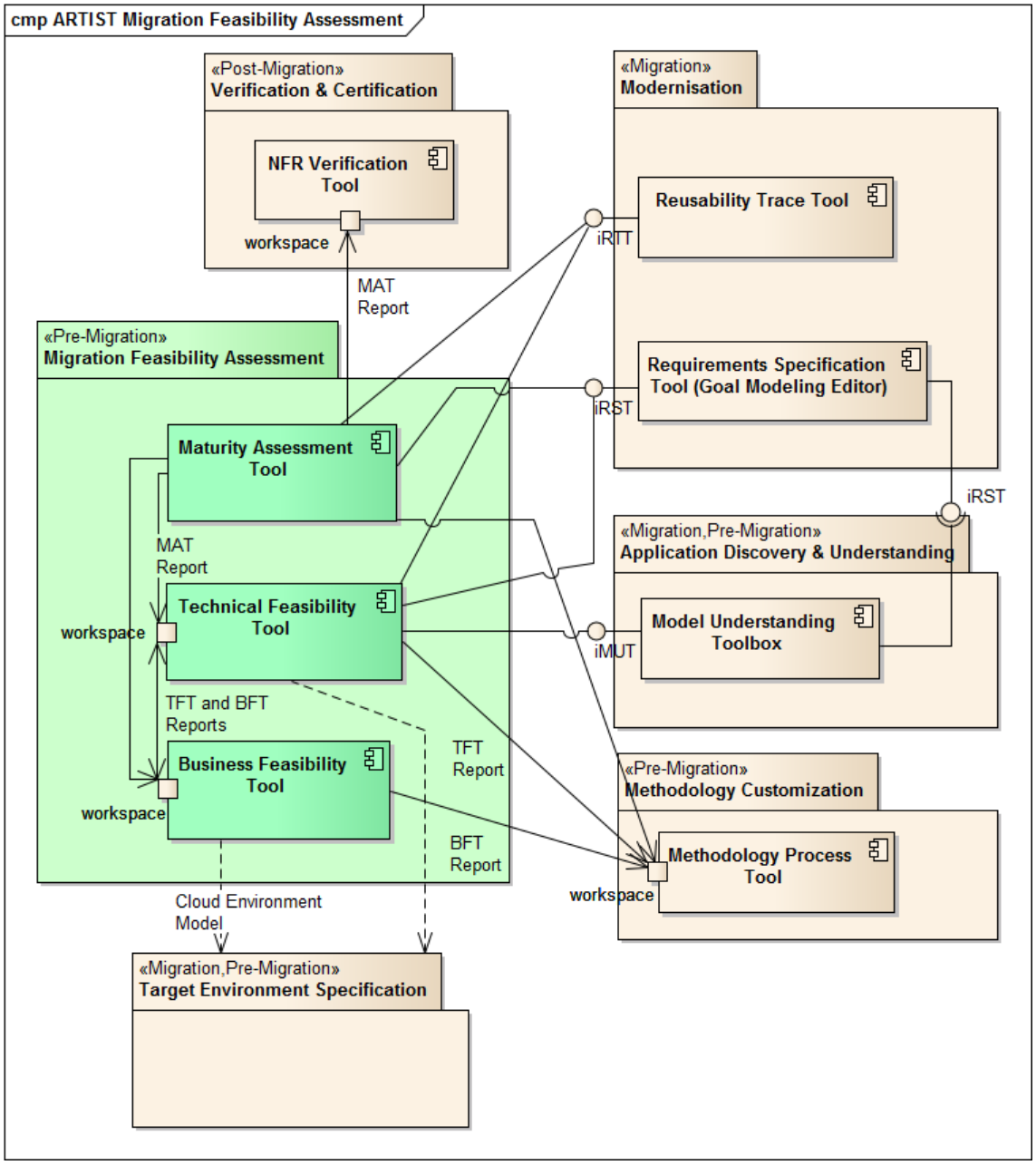


Figure 1 - ARTIST Migration Feasibility Assessment Package

2.1.2 Technical description

2.1.2.1 Prototype architecture

Next figure outlines TFT tool architecture design. M24 implementation implements new versions of the components, Software Complexity Component (SCC), TFT UI and also new versions of Effort Estimator and Strategy Suggestor. TFT supports partial integration with MAT reports and MUT high level models. Despite this report describes the complete TFT architecture for the M24 prototype.

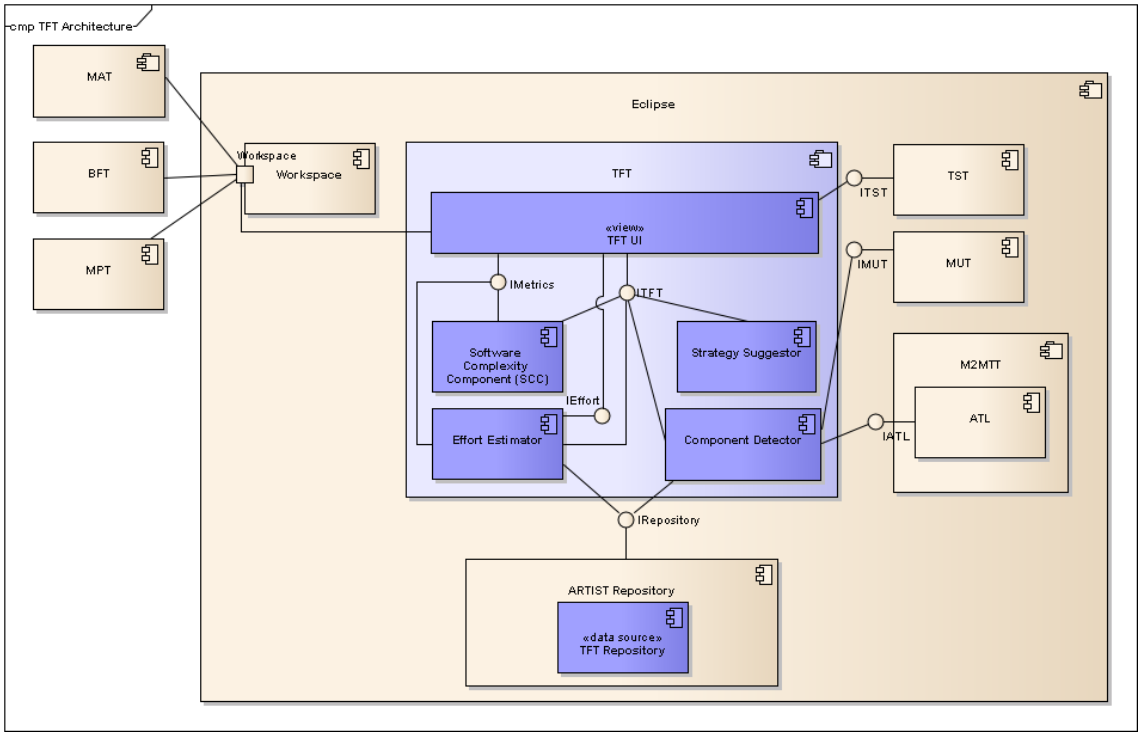


Figure 2 - TFT Architecture

TFT consists of a set of Eclipse elements (views, widgets, wizards), a set of backend components and a set of external dependencies; all of which are described below

TFT tool consists of:

- a set of Eclipse views and other widgets and wizards, which constitute the TFT UI and contributes to the Eclipse workbench through its exposed extension points. In this version (M24) of the prototype, Migration Efforts View has been introduced to this set of views. Among the other existing views, Inventory View and Migration Goals View have been improved in this version.
- a set of backend components, which provides some important TFT services, such as:
    - Component Detector
    - Software Complexity Component
    - Effort Estimator
    - Strategy Suggestor
    - TFT Repository (a bunch of TFT required artefacts stored in the ARTIST Repository)

  Component Detecor, Software Complexity Component and Effort Estimator have been improved significantly, the details of these improvements can be found in section 2.1.2.3. As a new back-end component, Strategy Suggestor has been improved in this version, the details of which can also be found in section 2.1.2.3.
- a set of external dependencies, notably other ARTIST components and tools, accessed through well-defined interfaces:
    - Maturity Assessment Tool (MAT)
    - Business Feasibility Tool (BFT)
    - Target Specification Tool (TST)
    - Model Understanding Tool (MUT)
    - M'M Transformation Tool (M2MTT)
    - Reusability Analysis Tool (RAT)

- ARTIST Repository
- Methodology Framework Tool (MFT)

There has been no significant updates on the interaction between TFT and these external dependencies except a better analysis is now carried out for MUT's output and a future work is planned for M30 to use MAT's new report in GML (Goal Modeling Language) format.

Next sections describe in more details TFT plugin components and dependencies.

2.1.2.2 TFT-UI

TFT offers its functionality through TFT UI, which the user can use to interact with TFT and its backend components. TFT UI contributes to the Eclipse workbench, complementing the Eclipse ARTIST perspective. The following views have been improved or newly developed in this M24 TFT UI release (see [D5.3.1] for detailed description of TFT UI views included in M12 TFT release):

- Inventory of components View: M24 version of this view has been updated with lots of changes and improvements upon existing functionality. The view now supports the display of nested components and multiple migration strategy suggestions per component. The migration strategy suggestion modifications are now done via a dialog where the user can see all the compatible and suggested strategies instead of a combo box. A report creation function and an option to open the new Migration Efforts View have also been added to the view.
- Migration Efforts View: This newly introduced view can be reached through the Inventory view, which displays detailed information about the component complexity, overall migration complexity and overall migration effort for each component.
  - Component complexity: The complexity of the component obtained from the SCC.
  - Overall migration complexity: This is the information of how complex would it be to perform all the suggested migration strategies for this component during the migration process.
  - Overall migration effort: The estimated effort for this component to be migrated to the target platform.

  The view also contains complexity and effort information for the migration of the all application. The user can also get detailed information on one component by double clicking the component's row. This user action brings front a dialog where complexity of the component, complexity values for each assigned strategy and effort estimations for each individual strategy is displayed.
- Migration Goals View: This version of the prototype's migration goals view is now fully responsive to user actions. Although the view does not allow the modification of the migration goals, the user is now able to select which migration goal should be taken into consideration for migration thus affecting the migration strategy suggestion process. This selection is done via checking or unchecking the checkboxes associated for each migration goal in the migration goals view. The goals can be mass-selected or deselected by clicking the checkbox of a parent node. The user can also modify the target platform selection through this view which is now reflected on the migration strategy suggestions, complexity and effort calculations. This modification on the target platform selection is done by using the target platform combo box provided in this view. The user is free to select any target platform that is supported by TFT from the list of platforms in the combo box.

- See [5] D5.3.1 - Technical Feasibility Tools for other views which belong to TFT-UI.

2.1.2.3 Components description

The following TFT backend components provide business logic support to the TFT UI:

Component Detector: In this version of the prototype, this component has been improved to take inter-component and parent child relationships between high level components into account for a better understanding of the migration candidate application.

Software Complexity Component: This component has been updated with several changes and improvements in this version of the prototype including new metrics calculation and supporting other technologies other than Java (C#). These updates are detailed in section 3 of the document.

Strategy Suggestor: This new component is responsible for analysing the components of the non-cloud compatible application and the relationships between these components and suggesting certain migration strategies for each component to assist the user in the pre-migration process. As the Component Detector, this component also relies on the high level component models generated by MUT in order to have a basis for the analysis of the application. Strategy Suggestor is also responsive to user feedback and dynamically analyses the model to update strategy suggestions.

Effort Estimator: This new component estimates the effort required to accomplish each required migration strategy suggested for each component that are marked for migration. This estimation is based on:

- The task type and complexity, which is calculated based on heuristics and the user range estimation (low, average, high)
- The complexity of affected component, which is estimated by the SCC.
- The number of reusability components for each task as well as the reusability level.
- Weighted factors obtained from heuristics

In this current version of the TFT prototype, Effort estimator is responsible for assigning effort weight values to each suggested migration strategy according to strategy complexity. These values are then used to calculate the estimated effort values for each component once SCC the metrics from SCC are obtained (See section 4).

TFT Repository: This component is not offered in this version of the prototype but planned to be available with the next deliverable in M30. This component will store historical data and heuristics required to estimate efforts. The M24 TFT implementation is local, that is, this content is shipped within TFT plugin upon installation from ARTIST update site. Incoming version will rely on the ARTIST repository to store and retrieve this information.

2.1.2.4 Technical Specifications

TFT, as an Eclipse plugin, is based on Eclipse RCP libraries and written in Java. TFT requires Java 6, a minimum Eclipse version of 4.3 (Kepler), and plugins EMF-Ecore, EMF-Query, Papyrus, UML2 installed.

User Interface

M24 version of TFT does not introduce a change in technology used within the user interface except a minor addition which introduces the usage of Eclipse dialogs for migration strategy selections in Inventory View and display of detailed information about complexity and effort

values in the Migration Efforts View. For functional changes and improvements on TFT UI, see section 2.1.2.2.

For technical specifications of TFT UI, see [5] D5.3.1 Technical Feasibility Tools

Back-End

Although there is no introduction of new technologies in this version (M24), the back-end of TFT has been significantly improved with regards to M12 version of the prototype. For the technologies used in the back-end of TFT see [5] D5.3.1 Technical Feasibility Tools

Software complexity component is now fully integrated to the other components to TFT enabling a more precise calculation of effort estimations for each component and migration strategy. In M12 version, Migration strategy definitions were hard coded into TFT the knowledge base. These definitions are moved to a an XML file which is parsed on TFT launch to populate a migration strategy container to be used by TFT components to ensure a more maintainable and modular implementation. The component model has also been extended to include dependency relationships among components and support nested component structure.

The knowledge base of TFT has been enhanced with more than 100 new rule addition handling complex suggestion scenarios and providing migration strategy suggestion support for target platforms; AWS, Windows Azure and Google App Engine. The knowledge base has also been improved to handle the update of migration strategy suggestions upon user modifications on existing suggestions.

An example to a migration strategy suggestion scenario from M24 version of TFT would be:

```
rule "IsStandardComponent"
 when
   $component : Component ( !applicationLevelComponent )
 then
   //SUPER RULE - DO NOTHING
end

rule "IsDataTypeComponent" extends "IsStandardComponent"
 when
   eval ($component.hasStereotype(Category.DATA.getPossibleStereotypes()))
 then
   //SUPER RULE - DO NOTHING
End

rule "Data_NOSQL" extends "IsDataTypeComponent" agenda-group  "GAE"
   when
     eval(mGoals.getTechnicalGoals().getA_p().getDatabaseScalabilityRequirements().equals(
"NO-SQL"))
   then
   MigrationStrategy strategy =
MigrationStrategyContainer.INSTANCE.getMigrationStrategy("HRD");
       strategy.setComplexity(Complexity.HIGH);
       strategy.setSuggested(true);
       $component.addMigrationStrategy(strategy);
end

rule "Data_RDBMS" extends "IsDataTypeComponent" agenda-group  "GAE"

   when
        eval(mGoals.getTechnicalGoals().getA_p().getDatabaseScalabilityRequirements() ==
        "RDBMSmultitenant")
   then
```

```
	MigrationStrategy strategy =
MigrationStrategyContainer.INSTANCE.getMigrationStrategy("CLOUD_SQL");
	strategy.setComplexity(Complexity.LOW);
	strategy.setSuggested(true);
	$component.addMigrationStrategy(strategy);
end
```

For a given component which is not marked as an application level component, the first rule IsStandardComponent returns true. Then a second rule IsDataTypeComponent which extends the first one gets fired and if this component has one of the stereotypes in the DATA category applied on it, the rule returns true. The second and third rules extend the IsDataTypeComponent rule so both of them get fired. The extending rules check the database scalability requirement of the migration goals. If the requirement is equal to NO-SQL, then the migration strategy is set to be HRD and complexity HIGH. If not and the requirement is set to RDBMSmultitenant then the migration strategy is set to CLOUD_SQL and complexity to LOW. The future work plan includes enhancing and improving these rules even more to be more comprehensive, making use of all the information we have on the legacy components in a smart way to decide which migration strategy and which level of complexity should be assigned to the component.

2.2 Delivery and usage

2.2.1 Package information

Delivery package consists of the below folder structure.

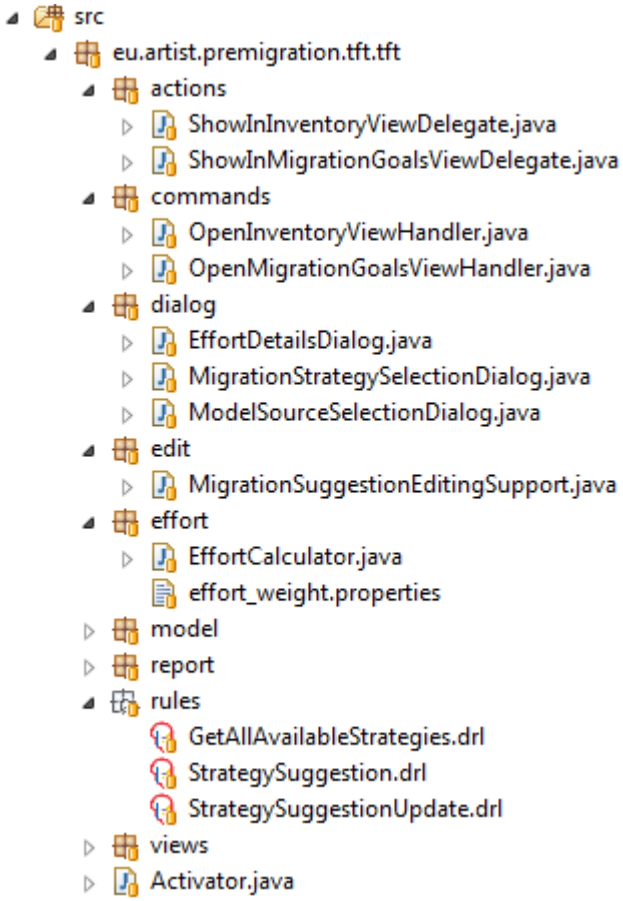


Figure 3 - Package structure of TFT

src: See [5] D5.3.1 - Technical Feasibility Tools.

eu.artist.tft.actions: See [5] D5.3.1 - Technical Feasibility Tools.

commands: See [5] D5.3.1 - Technical Feasibility Tools.

dialog: Contains the *MigrationStrategySelectionDialog.java* which handles the dialog operations for Migration Strategy selection for a specific component. The *EffortDetailsDialog.java* implements the dialog where the effort and complexity details for a component are shown to the user. *ModelSourceSelectionDialog.java* implements a dialog where the user is asked to select the source code paths for the component model. This source path information is used to estimate the migration effort fort each component.

edit: Contains the *MigrationSuggestionEditingSupport.java* which offers editing support for the Migration Strategy cells in the Inventory View table.

effort: Contains the *EffortCalculator.java* and *effort_weight.properties* which contains operations for effort calculations for Migration Strategies and effort weight values respectively.

model: This package contains Java models of UML components, Migration Goals: *BusinessGoals.java, Component.java, MigrationGoals.java, TechnicalGoals.java* and *MigrationGoalsContainer.java*. The sub-package query contains *ComponentModelQuery.java* class. This class has the necessary methods for querying the UML Component Diagrams. The migration strategy sub-package contains *Category.java*, *Complexity.java*, *MigrationStrategies.java*, *MigrationStrategy.java*, *MigrationStrategyContainer.java*, *MigrationStrategies.xml*, and *MigrationStrategies.xsd*. The classes in this sub-package are java class mappings of the migration strategies which are defined in the *MigrationStrategies.xml*.

rules: This package contains *StrategySuggestion.drl, StrategySuggestionUpdate.drl* and *GetAllAvailableStrategies.drl* Drools files. The first file contains all the necessary rules for the migration strategy suggestions in the Inventory view. The second file contains the rules that filter out the strategy suggestion combo box data from unrelated data. The last rule file contains the rules which fetch all compatible migration strategies for a specific component.

views: Contains the TFT views; *InventoryView.java, MigrationGoalsView.java* and *MigrationEffortsView.java*

models: Contains the usecase UML models, migration goals report and petstore UML model to be used for testing purposes.

profiles: Contains the UML profiles, which are applied to the models in the models folder.

lib: See [5] D5.3.1 - Technical Feasibility Tools.

META-INF: See [5] D5.3.1 - Technical Feasibility Tools.

plugin.xml: See [5] D5.3.1 - Technical Feasibility Tools.

2.2.2 Installation instructions

TFT plugin will be available for installation using Eclipse's Install New Software feature via ARTIST's update site in the future. Now, it can be downloaded and installed manually. Eclipse 4.2 (Juno) is required for installation but Eclipse 4.3 (Kepler) is recommended.

Installation Procedure of Required Plugins

EMF-Ecore, EMF-Query, Papyrus, UML2 plugins are required to be installed. All these plugins may be installed by following the installation procedure below:

1. On the Help menu of Eclipse, click Install New Software.
2. From the "Work with" dropdown list select Kepler or Juno depending on your Eclipse version. If you don't have these entries in the list continue on from step 2a otherwise continue from step 3.
    a. Click Add button at the right side of the Work with dropdown list.
    b. Write a name of your choice for the Kepler/Juno repo.
    c. Write http.download.eclipse.org/releases/kepler or http://download.eclipse.org/releases/juno to the Location field and click Ok.
3. Select the plugin(s) you want to install (EMF Core, EMF Query, UML2 and Papyrus UML) click next and follow the instructions on screen.
4. Restart Eclipse.

TFT Installation Procedure

1. Download eu.artist.premigration.tft.tft.jar and eu.artist.premigration.tft.scc.jar files from ARTIST repository.
2. Copy these jar files to Eclipse's dropins folder.
3. Restart Eclipse

2.2.3 User manual

A general view of the Eclipse Workbench with TFT views and views that interact with TFT open is below.

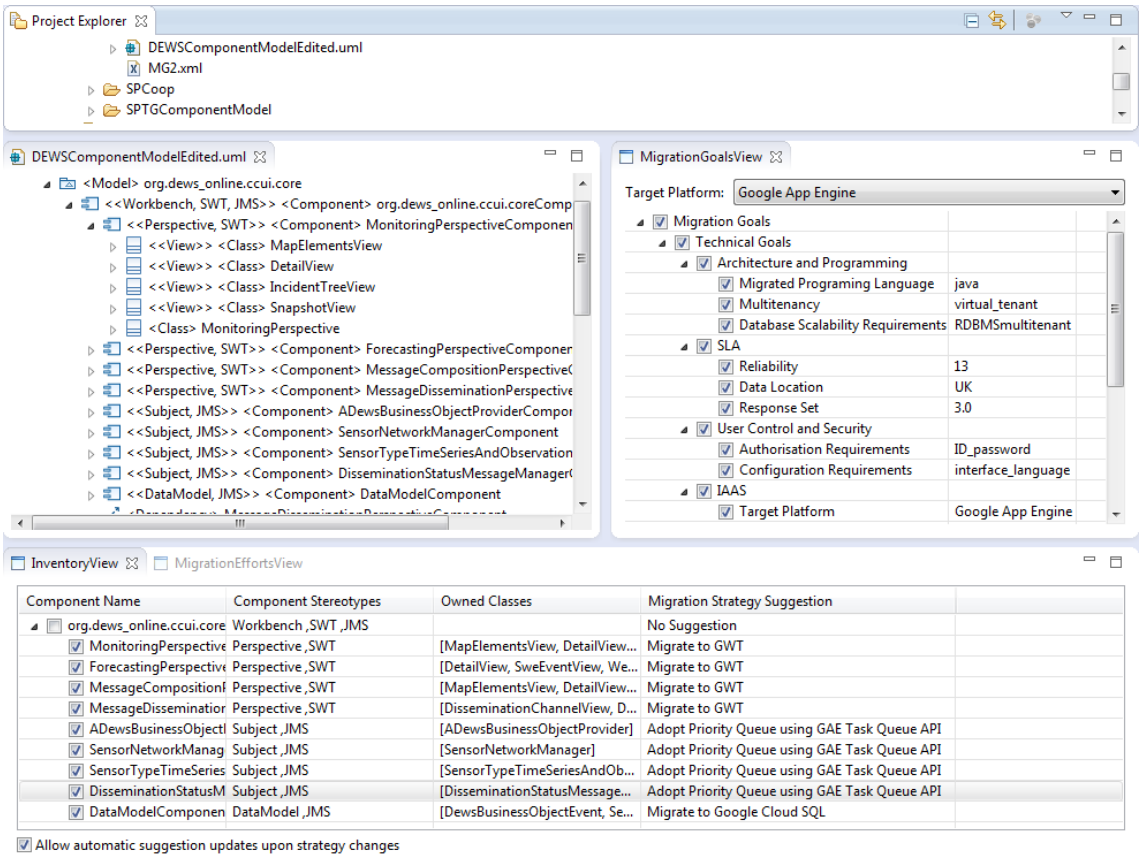


Figure 4 - Eclipse workbench with TFT views

At the top pane, a sample test project containing a JEE6 Component model diagram of the legacy application is open. By right clicking the Papyrus container, the DI or the UML file, the Inventory view can be opened. Also, MG2.xml can be seen inside the testing project. This MAT report file is used to open the Migration Goals View which can be seen on the upper right region of Figure 4: Eclipse workbench with TFT-TFT views.

On the middle-left pane, a UML Editor is open. The user is free to modify the model on this or another UML editor of her/his choice (e.g. Papyrus). The model with the new modifications is analysed and suggestions are updated once the user saves the changed model. On the right of this pane the Migration Goals View can be seen. On the bottom page, two tabs can be seen, Inventory View and the Migration Efforts View. These three views will be explained in detail in the following sections.

2.2.3.1 Inventory view

In order to open the Inventory View, the user must first open the Migration Goals View since the migration tasks to be suggested in the Inventory View depends on the migration goals. If the user tries to open the Inventory View before the Migration Goals View, a warning message pops up, informing the user about this, and asks for Migration Goals View to be opened first.

To open the inventory view of TFT plugin, the user should right click the component diagram UML file which is the output of model discovery process and select Show in Inventory View menu action. This action can also be found in the context menus of the files of type .di and Papyrus Model Containers.

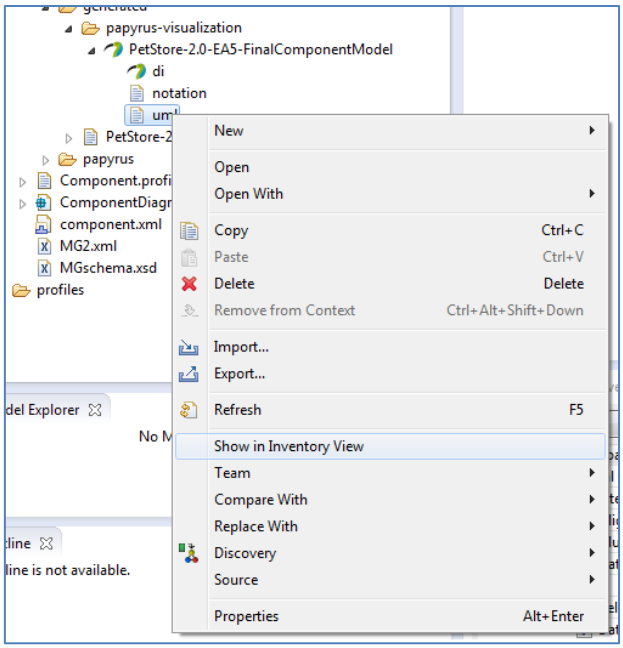


Figure 5 - Opening a model in Inventory View

After selecting the "Show in Inventory View" menu option, the user will see the dialog below asking the source file location(s) of the selected component model.

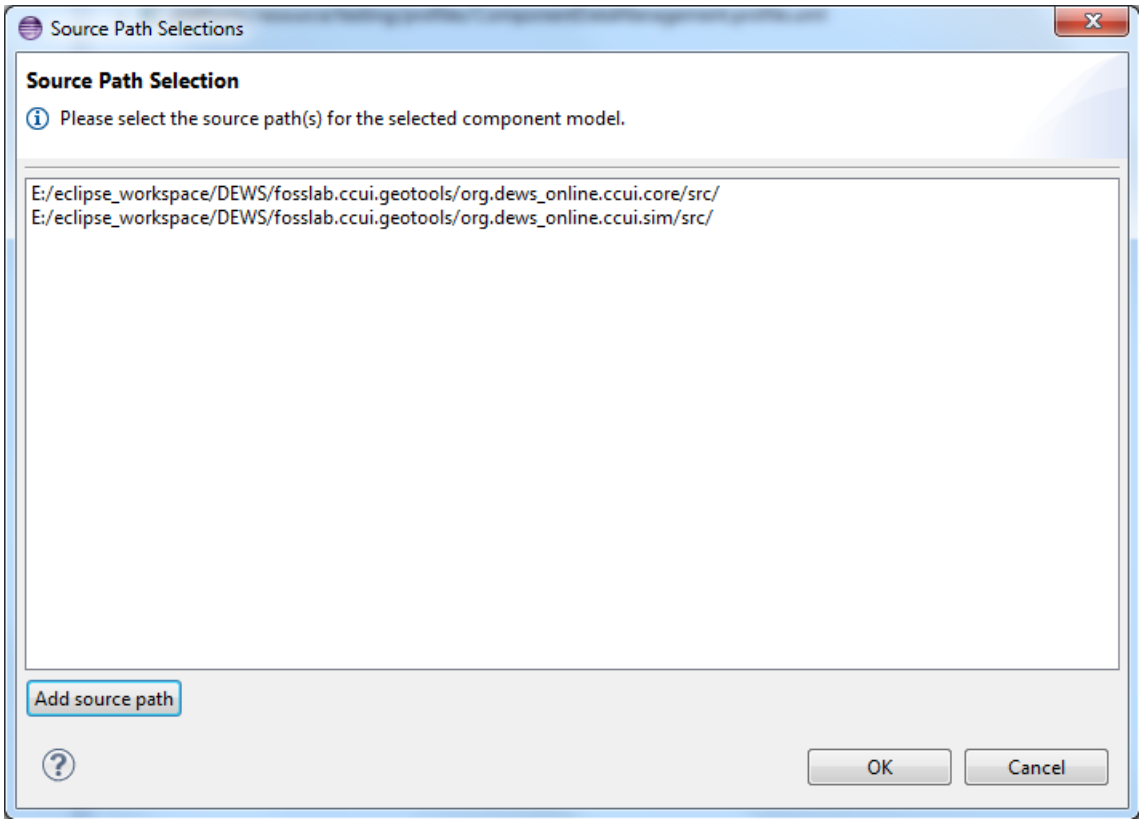
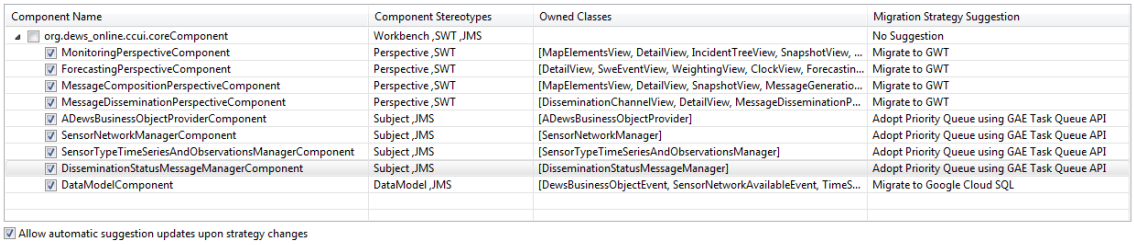


Figure 6 - Source path selections dialog

Using this dialog's "Add source path" button the user should locate the source folders containing the classes referred by the component model then press OK to proceed and open the inventory view.

The inventory view consists of a table which has the components of the selected component diagram. Component name, stereotypes, the classes owned by the component and the migration strategies suggested for the component by TFT can be seen in each row.



Component Name	Component Stereotypes	Owned Classes	Migration Strategy Suggestion
org.dews_online.ccui.coreComponent	Workbench ,SWT ,JMS		No Suggestion
MonitoringPerspectiveComponent	Perspective ,SWT	[MapElementsView, DetailView, IncidentTreeView, SnapshotView, ...	Migrate to GWT
ForecastingPerspectiveComponent	Perspective ,SWT	[DetailView, SweEventView, WeightingView, ClockView, Forecastin...	Migrate to GWT
MessageCompositionPerspectiveComponent	Perspective ,SWT	[MapElementsView, DetailView, SnapshotView, MessageGeneratio...	Migrate to GWT
MessageDisseminationPerspectiveComponent	Perspective ,SWT	[DisseminationChannelView, DetailView, MessageDisseminationP...	Migrate to GWT
ADewsBusinessObjectProviderComponent	Subject ,JMS	[ADewsBusinessObjectProvider]	Adopt Priority Queue using GAE Task Queue API
SensorNetworkManagerComponent	Subject ,JMS	[SensorNetworkManager]	Adopt Priority Queue using GAE Task Queue API
SensorTypeTimeSeriesAndObservationsManagerComponent	Subject ,JMS	[SensorTypeTimeSeriesAndObservationsManager]	Adopt Priority Queue using GAE Task Queue API
DisseminationStatusMessageManagerComponent	Subject ,JMS	[DisseminationStatusMessageManager]	Adopt Priority Queue using GAE Task Queue API
DataModelComponent	DataModel ,JMS	[DewsBusinessObjectEvent, SensorNetworkAvailableEvent, TimeS...	Migrate to Google Cloud SQL

Allow automatic suggestion updates upon strategy changes

Figure 7 - Inventory View

The user is given the option to change the suggested migration strategy for a specific component. This is done by clicking the Migration Strategy Suggestion cell of the component in the table and clicking the ... button on the right-side of the cell. This action opens the Migration Strategy Selection Dialog which is described in detail in the next section.

Inventory view is fully responsive to the changes on the migration strategy selections and also to the changes on the component model. If the user modifies migration strategy selections for a component via the Migration Strategy Selection Dialog and clicks ok, the rule engine steps in and refreshes the migration strategy suggestions for all components. This automatic suggestion update can be disabled using the checkbox below the Inventory View table. Also, if the user decides to add a component, or changes a component's properties by using the component editor, the inventory view is refreshed and the strategy suggestions are recalculated when the changed model is saved.

2.2.3.2 Migration strategy selection dialog

The migration Strategy Selection Dialog opens when the user selects the migration strategy cell for a component in the Inventory View and clicks the ... button. In this dialog, the user is free to change the migration strategy suggested by TFT, remove strategies and add new ones. The dialog allows only one strategy per category.

For example, if the component has both Data and Framework related features, two migration strategies are suggested; Migrate to Google Cloud SQL and Migrate to Spring. In order to add another Data related migration strategy, let's say Migrate to HRD, the user first has to remove the Migrate to Google Cloud SQL strategy from the Selected Migration Strategies list.

If the selected migration strategies are modified, the user is warned about the possibility of the update on the migration strategy suggestions of the components that has a dependency relationship with the component selected, after the Ok button is clicked.

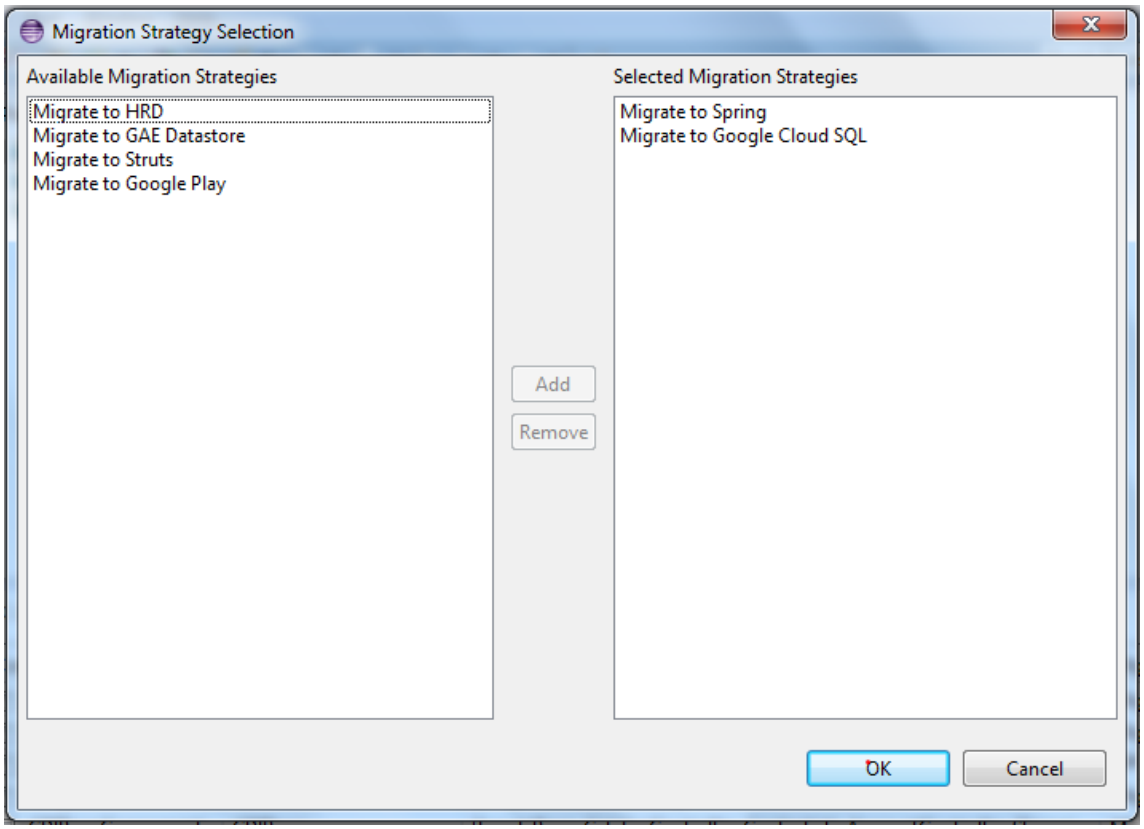


Figure 8 - Migration Strategy Selection Dialog

2.2.3.3 Migration efforts view

To open the Migration Efforts View the user should click on the Open Efforts View button provided in the Inventory View. This view contains a table where for each component, the component complexity, overall migration complexity and overall migration effort is shown (Figure 8: Migration Efforts View). The user can double click or right click a component and select show details menu option to bring forth a dialog where complexity information for each migration strategy suggested for this component can be seen.

Component Name	Component Complexity	Overall Migration Complexity	Overall Migration Effort
org.dews_online.ccui.coreComponent			
MonitoringPerspectiveComponent	0.54	Average	3.81
ForecastingPerspectiveComponent	0.57	Average	4.00
MessageCompositionPerspectiveComponent	0.60	Average	4.21
MessageDisseminationPerspectiveComponent	0.63	Average	4.41
ADewsBusinessObjectProviderComponent	0.43	Average	2.14
SensorNetworkManagerComponent	0.51	Average	2.54
SensorTypeTimeSeriesAndObservationsManagerComponent	0.67	Average	3.33
DisseminationStatusMessageManagerComponent	0.54	Average	2.71
DataModelComponent	0.37	Low	2.21

Figure 9 - Migration Efforts View

2.2.3.4 Migration goals view

To open the Migration Goals View of TFT plugin, the user should right click the xml file containing the migration goals which is generated by the MAT and select Show in Migration Goals View menu action.

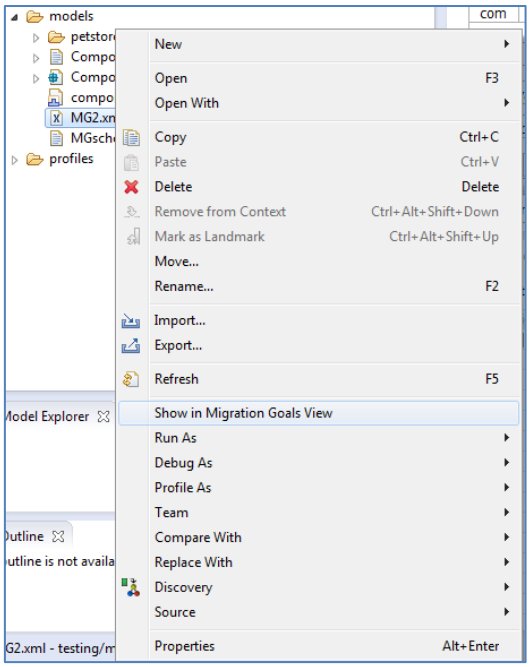


Figure 10 - Opening the migration goals view

The Migration Goals view consists of a combo box and a tree table.

- The combo box is filled with supported target cloud platforms with regards to the MAT report and default value is selected with respect to the value of the target platform in the migration goals. The combo box value can be changed by the user, which surely affects the suggested migration strategies in the Inventory View.

- The tree table lists the migration goals in a parent – child relationship with their respective values. The user has no control over these values but can enable and disable the goals by clicking the checkboxes. In case the user wants to edit the values of the migration goals, he/she has to start over the MAT questionnaire again and go through the same process since these goals are determined by MAT based on the user's responses. Enabling/disabling the goals also affects the migration strategy suggestions in the Inventory View.

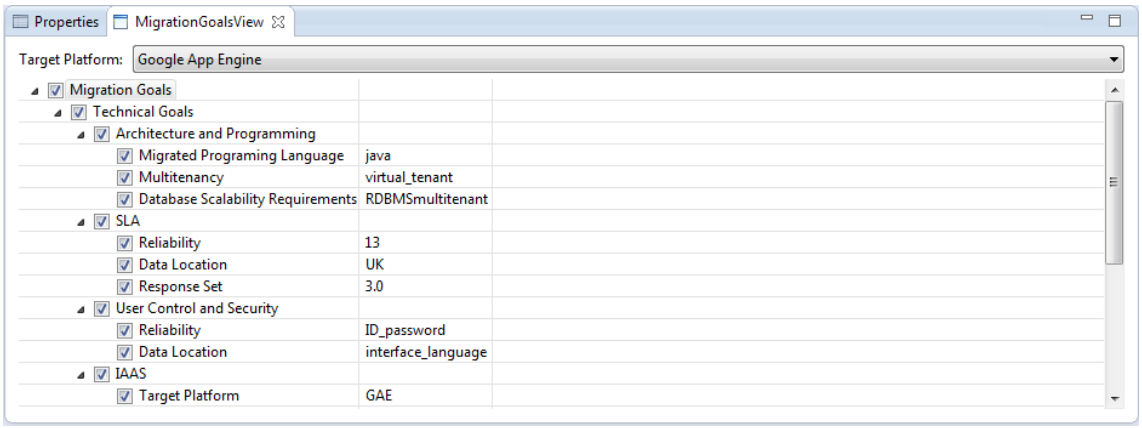


Figure 11 - Migration Goals View

2.2.4 Licensing information

This component is offered under EPL v1.0 license.

2.2.5 Download

Source code for this version of the TFT prototype may be viewed\downloaded from the ARTIST Tooling Github repository, here.

3 SCC

3.1 Implementation

The Software Complexity Component (SCC) aims to provide a new set of metrics in the technical feasibility phase prior to the migration. Thus, the results from metrics analysis can provide a better insight whether migration should be feasible or not.

In the first version of the prototype delivered in M12, the Level of Complexity (LOC) metric was provided for each Java class.

In the current version of the prototype, at M24, a set of new metrics are provided. These metrics allow the calculation of the Modifiability, Understability and Scalability (partially) metrics [1], which in combination with the LOC metric (provided in the previous version of the prototype), support the calculation of the final metric to be provided by the SCC, the Maintainability metric.

Furthermore, in the current version of the SCC, metrics for C# based code are also supported. So the input to the SCC can be source code from an application written in any of the two technologies supported by the ARTIST project.

The metrics provided by the current version of the prototype are those needed to calculate the Level of Complexity (already provided in the M12 prototype) the Modifiability, Understability and partially the Scalability. The new atomic metrics calculated by the M24 prototype are:

- NGen: Number of generalizations: The Number of Generalization metric is defined as the total number of generalization relationships within a class diagram (each parent-child pair in a generalization relationship).
- NAggH: Number of aggregation Hierarchies: The Number of Aggregation Hierarchies metric is defined as the total number of aggregation hierarchies within a class diagram. The Number of Aggregation metric is defined as the total number of aggregation relationships within a class diagram (each whole-part pair in an aggregation relationship).
- MaxDIT: Maximum Depth of Inheritance. The Maximum DIT metric is defined as the maximum between the DIT value obtained for each class of the class diagram. The DIT value for a class within a generalization hierarchy is the length of the longest path from the class to the root of the hierarchy.

These metrics are calculated per component (defined in the component model by the MUT).

3.1.1 Fitting into overall ARTIST solution

SCC prototype is part of the Technical Feasibility Tool. In the first version of the prototype, SCC and TFT were provided as separate components, but in the current version of the prototype, they are provided as a single tool.

TFT (and thus SCC) is the tool that ARTIST project provides to perform the Technical Feasibility analysis proposed in pre-migration phase of the ARTIST methodology (see Figure 1).

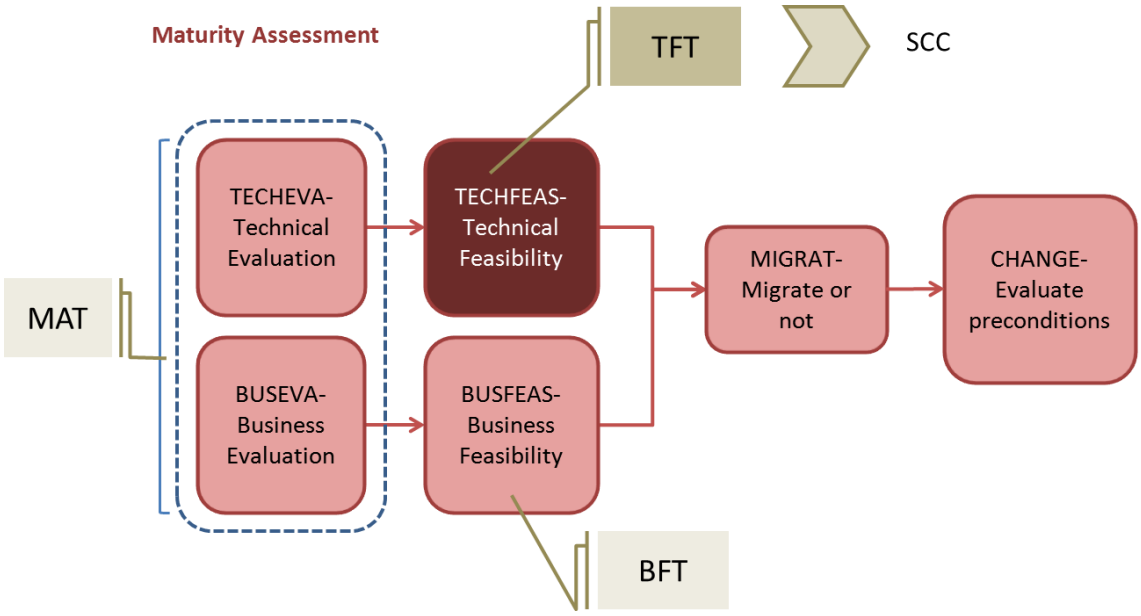


Figure 12 - SCC in overall Maturity Assessment process

SCC as part of the TFT is also part of the modernization assessment package (see section 2.1.1).

3.1.2 Technical description

3.1.2.1 Prototype Architecture

The current SCC prototype architecture is a Java API that explores source files an UML models to generate several metrics of a specific project. The following image depicts the overall architecture:

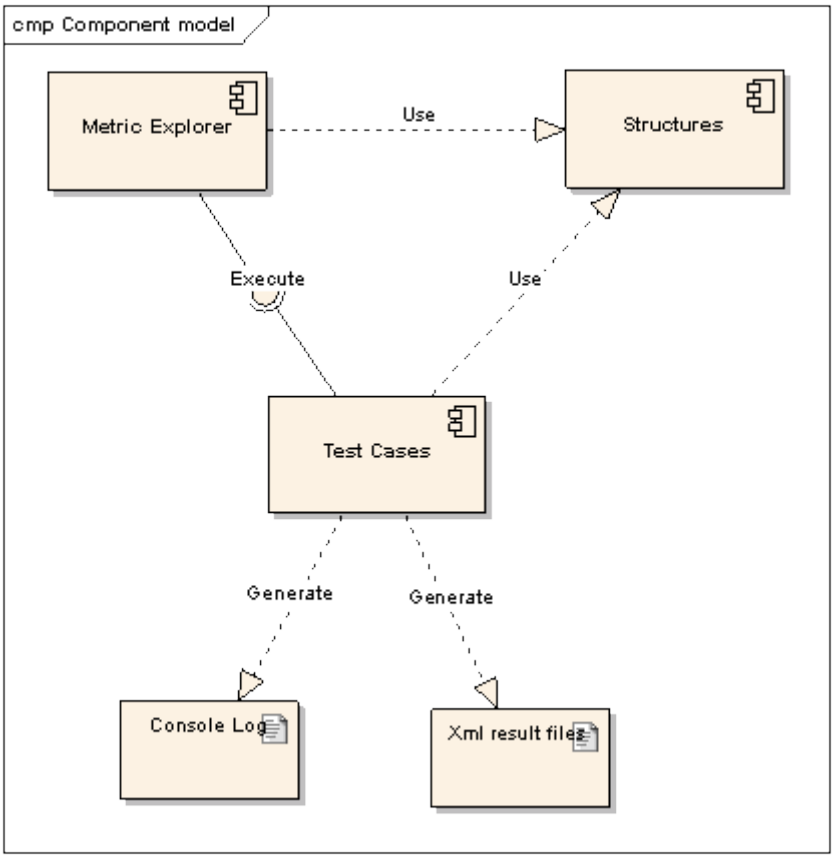


Figure 13 - SCC high level architecture

While the main objective of the Artist Metrics Generator is to expose an API that any other plug-in or RCP could use to obtain the metrics generated in the Artist project, it also provides Test Cases to access the same functionality as if used programmatically. The generated metrics are available in XML files and console log.

3.1.2.2 Components description

The current SCC prototype component comprises three components:

- **Metric Explorer:** This is the main component of SCC current prototype. It provides the calculation of all the required metrics that are used to generate the new ARTIST metrics. Besides, it also provides exporting features to convenient formats like XML or JSON**.**
- **Structures:** This component contains the structures of the inputs and outputs models that the Metric Explorer uses. It also provides the functionality for generating the output file formats (XML, JSON).
- **Test Cases:** This component is provided for implementing the testing of the Metric Explorer component. It generates several use cases that test the functionality of the SCC. The test case generates console logs and XML files with several examples (DEWS and JavaPetStore).

3.1.2.3 Technical specifications

All the components are developed in JavaSE 1.6. So this is the minimum java version for executing the API. There are no any other requirements.

User Interface

There is no user interface implemented as the result obtained from SCC is to be consumed by TFT-TFT and used for the required effort calculation (see section 4). For executing the Metrics Explorer API the user has to execute the test cases included in the API. Several input parameters can be changed for obtaining new metrics of different projects.

Back-end

There is no persistency implemented neither planned for this API. Thus, every time the new metrics are needed, they have to be calculated. Persistency is delegated to API consumers.

3.2 Delivery and usage

3.2.1 Package information

The following image depicts the package structure of the main component, the Artist metrics generator plug-in.

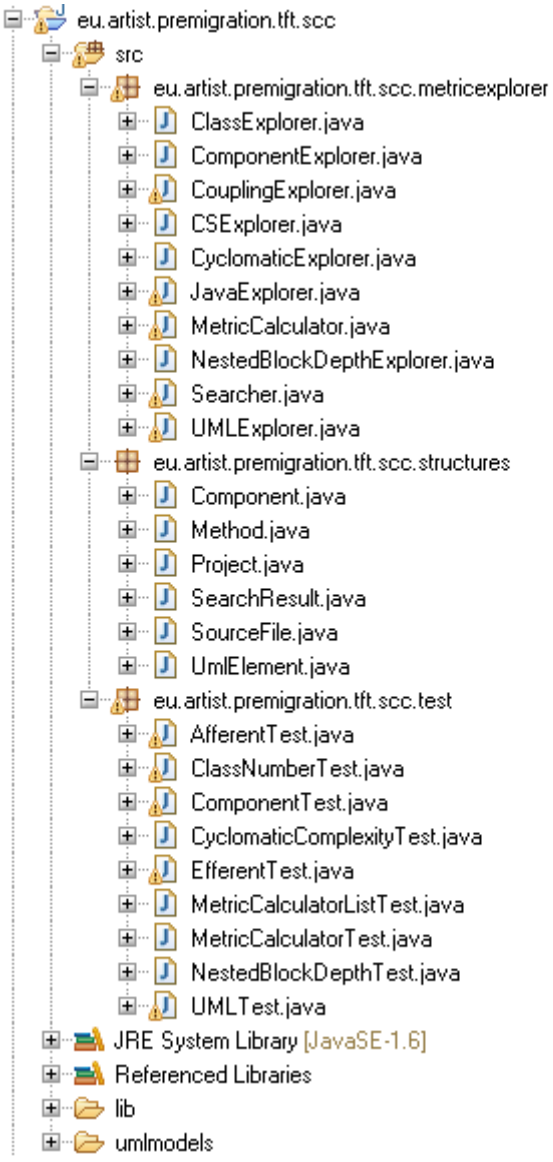


Figure 14 - Package structure of the ARTIST metrics plugin

- **eu.artist.premigration.tft.scc.metricexplorer**: Contains the classes for exploring the UML models and the source code.
- **eu.artist.premigration.tft.scc.strucctures**: Contains the classes of the structures used by the metric explorer component
- **eu.artist.premigration.tft.scc.test**: Contains the test cases classes for executing the metric explorer component.
- **umlmodels**: This folder contains the UML models used in the test cases and other UML models also.

3.2.2 Installation instructions

In this version of the prototype this plug-in requires manual installation. The user has to import the components to the Eclipse workspace manually.

3.2.2.1 Requirements

All the components are developed in JavaSE 1.6. So this is the minimum Java version for executing the API.

3.2.3 User manual

Import the project into the Eclipse workspace:

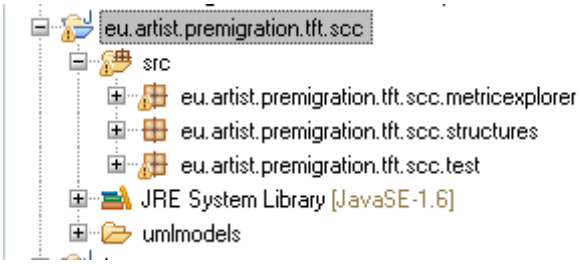


Figure 15 - SCC project

Open the eu.artist.premigration.tft.scc.test package

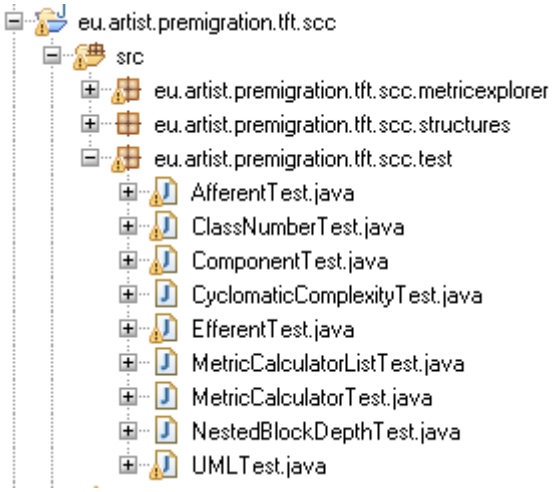


Figure 16 - Testing package

Right click in a test case and select the Run as Java Application option:

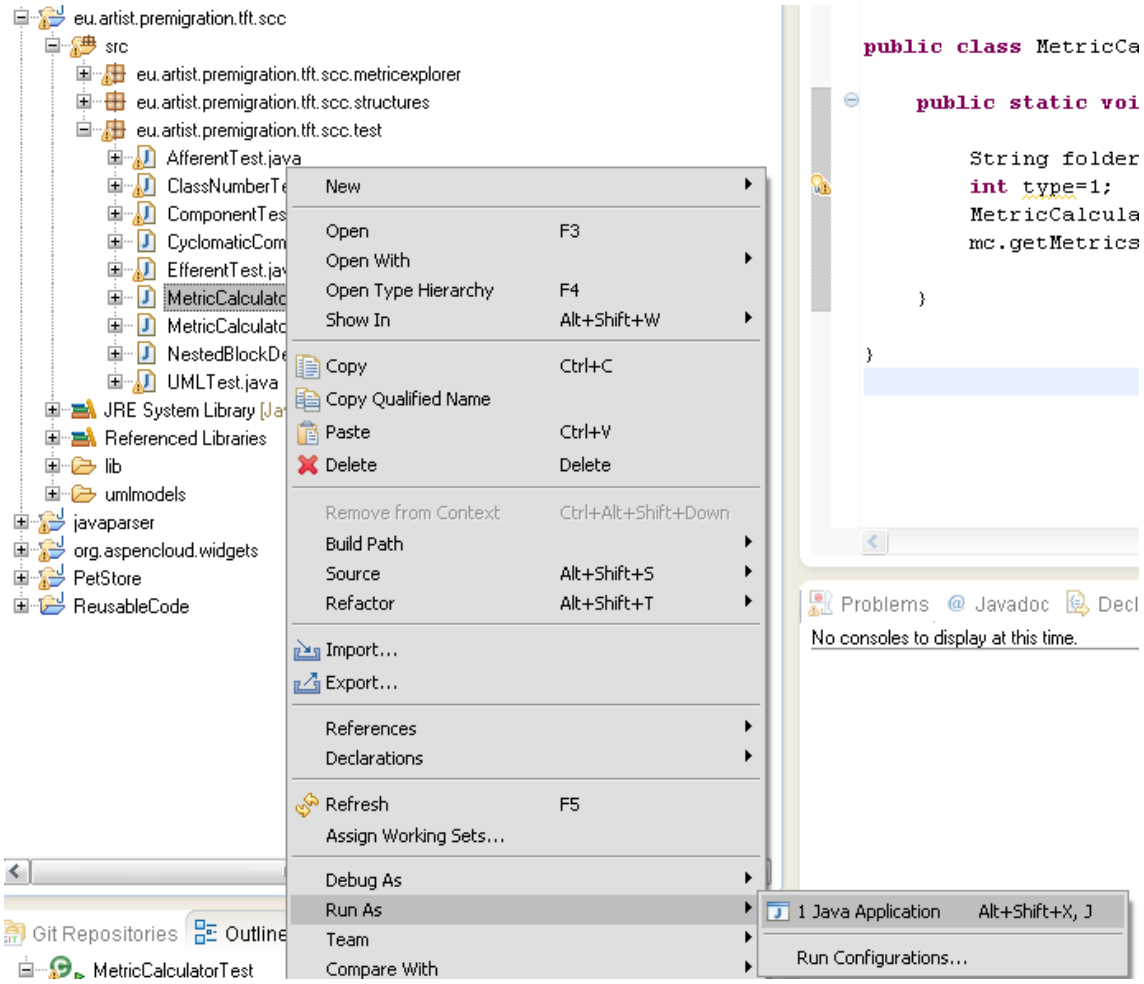


Figure 17 - SCC testing

***Note**: The user has to change the "hardcoded" input parameters of the test cases manually*

3.2.4 Licensing information

This component is offered under EPL license.

3.2.5 Download

This second release of SCC is available in the ARTIST github, more precisely at the following address:

https://github.com/artist-project/ARTIST-Tooling/pre-migration/technicalfeasibilitytool/SCC

4 Interaction of TFT components

The two components of TFT interact with each other in order to present the user an effort estimation of the selected migration strategies for each component. Figure 4 depicts an overall picture of this interaction.

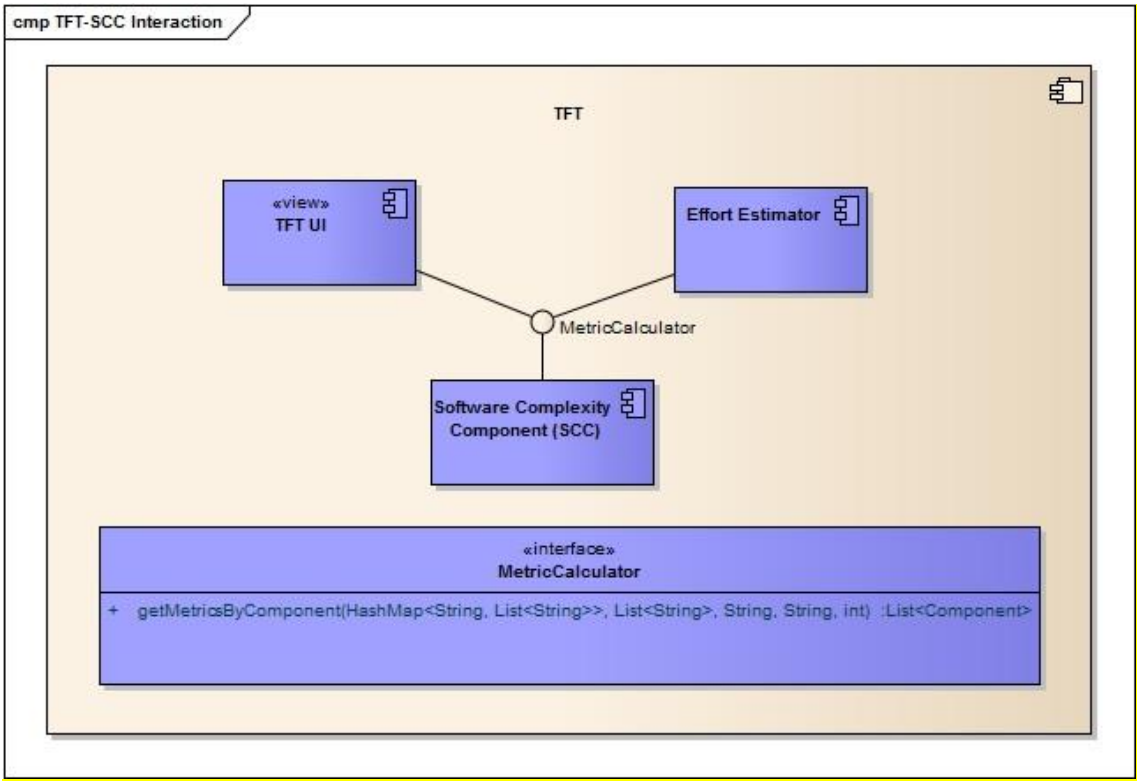


Figure 18 - Interaction of TFT components

SCC offers the MetricCalculator class containing the *getMetricsByComponent* method to be used by TFT to obtain the metrics calculated by SCC. The method signature can be found below.

- `getMetricsByComponent(HashMap<String, List<String>> classFiles, List<String> srcPaths, String ComponentModelPath, String ClassModelPath, int type)`

This method requires a map which has the component name as the key and the class files contained by this component as the value, a list of source paths which the source files can be found in, the path of the component model, the path of the class model and the source type (i.e. java, c# etc.). The method returns a list of components (`eu.artist.premigration.tft.scc.structures.Component`), each containing the metric information of the TFT components that are included in the component model of the migration candidate (`eu.artist.premigration.tft.tft.model.Component`).

TFT uses the metric information and the migration strategy complexities to calculate the estimated effort for each component's migration towards cloud. The calculated effort estimation is displayed in the Migration Efforts View which can be accessed from the Inventory View.

5 Conclusions

This document described the functionality, architecture, delivery method and installation/usage details of Technical Feasibility Tools, detailed the SCC component and the interaction between TFT components. The second version of TFT prototype containing several new features has also been provided.

In this version of the deliverable, SCC component introduced the Maintainability, Modifiability Metrics and the first design of the API that TFT will use to obtain the metrics from SCC. A baseline for the migration task effort estimation computation which will make use of the aforementioned metrics SCC will provide has been implemented. This information is shown on the Migration Efforts View along with component complexity and the migration strategies' complexities in a range of Low, Average and High. The combo boxes that allowed the user to select migration strategies has been switched with an Eclipse dialog which enables multiple strategy selections in a more user friendly way. Automatic migration strategy updates have been introduced which updates the migration strategies suggested for components which have a dependency relationship with the ones user has modified their migration strategy. The knowledge base of TFT has also been enlarged with new rules and new migration strategies derived from work package 9 [4].

In the next iteration, the focus will be on enhancing the knowledge base of rules supporting more complex scenarios and more specialized migration strategies, completing the implementation of the interaction between TFT's components and adding a support for MAT's new report format in GML.

6 References

[1] D5.1.1 – Specification of the Business and Technical Modernization Assessment in ARTIST

[2] D6.2.1 – ARTIST – Methodology

[3] D8.3 – ARTIST - D8.3 Methodology and techniques for model understanding M18

[4] WD9.4 – ARTIST – WD9.4 Collection of Optimization Patterns

[5] D5.3.1 - Technical Feasibility Tools

[6] D6.4.1 - ARTIST Integrated Architecture

7 Appendix: TFT knowledge base (library of rules)

7.1 Migration Strategy Taxonomy

In order to migrate a system to the cloud, certain tasks have to be fulfilled. These tasks may require modifications on the system itself and definitely some work has to be done on the target platform as well. These tasks can be organized within 4 main categories, Installation & Configuration, Modification, Migration and finally, Testing, which consist of more precise, sub-categories.

Installation and Configuration

Software installation and/or configuration may be needed locally or on cloud, for IaaS migrations, depending on the candidate system for migration before starting the code modification. The installations and configurations may be required or may just be a case of "nice to have".

An example for a requirement would be the case where a Java application, which uses JPA, is to be migrated to the cloud with SimpleDB. In this situation a third party library, SimpleJPA has to be installed and configured.

Although packaging and deployment of the application are in the context of the Migration category, some installation and/or configuration might be needed to automate/ease some parts of the migration process. For example: Azure plugin for Visual Studio must be installed in order to benefit from its automated packaging feature.

Modification

Modification category focuses on the required modifications on the system before migration and consists of 3 sub-categories: Database Changes, Code Changes and Connection Changes. The scope of modification depends on many parameters; target platform, the modules that are planned to be migrated or not, compatibility of the local and the target platforms etc.

- **Database changes:** If the database is also planned to be migrated to the cloud, it is very likely that some modification will be needed. Database scripts and schema modifications might be necessary in following situations:
    - Difference of database versions
    - Difference of database variants (MySQL vs MSSQL)
    - Difference of database types (Relational vs NoSQL)
- **Code changes:** If migration goals include enhancing overall performance, switching to more contemporary technologies etc., changes on the written code would be inevitable. Also if there has been a modification on the database, code change would probably be needed.
    - Database access layers, models may need to be rewritten. An example would be JPA to SimpleJPA conversion.
    - In a case of an MVC application migration, CDI Managed beans and controllers might be modified to integrate with Spring.

- **Connection changes:** Typically, two components of a system are connected with a LAN connection before the migration. According to the migration strategy, this connection may change to a WAN connection or a LAN connection in cloud. For either case, modifications below might be necessary,
    - Protocol optimizations to ensure data transfer efficiency
    - Security optimizations. If there will be a switch from LAN to WAN, some additional threat scenarios should be considered.

Migration

Being the last step before testing, this category defines the process of configuration and preparation of the modules targeted for migration and should be trivial if the previous steps are handled properly.

- **Configuration:** Application configuration files should be change\modified to handle the modifications done on the code in the previous category and also according to the target platform requirements.

  Additionally, some configurations on the project nature might be necessary in order to make use of some third party plugins. For example, Azure plugin for Visual Studio automates package and configuration file creation process, but in order to use the plugin, the project to be migrated must be a “Web Application Project”. If it is a “Web Site Project”, it has to be converted to a Web Application Project first.

- **Preparation:** In order to deploy the application and/or the database to the cloud some certain preparations may be required.
    - Preparation of the database: If the scripts had not been changed in the code modification step, some configuration changes might be necessary in order to use third party tools for DB migration automation.
    - Preparation of the application: Before the deployment, the application has to be packaged. In case of migrating to PaaS application may be required to be packaged according to a certain format.

Testing

This one of the most important - if not the most important - part of the migration as a whole. A successful deployment to cloud (migration process) is a proof of some of the actions taken are not erroneous; configuration changes, installations on IaaS etc. but this does not ensure that the application would run as expected.

Testing process should be well planned and carried out carefully. One example procedure for testing might be as follows:

- Testing if database migration to cloud is successful.
- Testing the application in local servers can communicate with the DB in Cloud.
- Testing if the migration of the application to cloud is successful
- Testing if the application in cloud can communicate with the DB in cloud.

- Carrying out functional, performance, security etc. tests

7.2 Migration strategies

Adopt Cloud Watch: This migration strategy suggests to adopt the Cloud Watch Service offered by Amazon EC2. Amazon CloudWatch is a monitoring service offered by Amazon that controls all the time the use of cloud infrastructure resources and allows system administrators to establish scaling rules based on the values of certain metrics.

Migrate to App Engine Datastore: This strategy suggests that necessary changes should be made on the component so that it becomes compatible with Google App Engine's Datastore. This service provides a schema-less object datastore including a query engine, atomic transactions support via the Datastore API. Its primary data repository (High Replication Datastore) replicates its stored data in multiple data centers providing high read/write availability and durability. Datastore is scalable in terms of reads and writes. Writes scale by distributing write data when necessary while reads scale by only permitting queries whose performance depends solely on the result set. Query serving uses pre-built indexes which impose restrictions on the supported operations over the stored data.

Migrate to Task Queue API: This strategy suggests that the related component should adopt and use the Google App Engine's Task Queue API (TQA). TQA allows applications to perform work outside of a user request, initiated by a user request. The work is divided into separate tasks and these task are pushed to task queues in order to be processed in aggregate.

Adopt Amazon Direct Connect: This migration strategy suggests adopting Amazon Direct Connect. This service offered by Amazon, enables the establishment of a dedicated network connection between AWS and other resources. It serves for keeping constant network latency needed in case of real-time data feeds as well as for utilize Amazon's infrastructure together with other infrastructure controlled by the user.

Test Database Connections: This migration strategy suggests testing the database connections after migration. Testing of the connection between the local servers and the DB in the cloud and the connection between the application in the cloud and the DB in the cloud should be carried out.