ARTIST Technical Feasibility Tool: Supporting the Early Technical Feasibility Assessment of Application Cloudifications

An approach for estimating the complexity of a cloudification project in a pre-modernization stage

Juncal Alonso Ibarra, Leire Orue-Echevarria, Zurik Corera Seoane
ICT-European Software Institute Division,
Tecnalia
Zamudio, Spain
juncal.alonso@tecnalia.com, leire.orue-echevarria@tecnalia.com, zurik.corera@tecnalia.com

Jesus Gorroñogoitia, Burak Karaboga
Atos Research & Innovation
Madrid, Spain
jesus.gorronogoitia@atos.net, burak.karaboga@atos.net

***Abstract*— Modernizing an IT system is a long, complex journey. The pre-migration phase is the starting point of each migration project where the decision to transform the legacy rather than to rewrite it has to be taken. In order to support this decision making, the ARTIST European project [1] proposes a technical feasibility analysis to as much technical information as possible about the legacy application itself and about the required technical tasks to migrate its components. This paper presents a technical feasibility analysis which relies on Cloud Migration Point approach to estimate the cost of the migration (in terms of required effort) and incorporates techniques such as Model Driven Reverse Engineering, software complexity metrics or Domain Specific Language-based heuristics to automate this process as much as possible, although leaving to the user the knowledge and control all over the entire process**

Keywords-Software modernization, technical feasibility, software complexity, cloud computing, migration strategy.

I. INTRODUCTION

Prior to facing a challenging project such as a software migration one, which may involve not only changing the way companies will deliver their software but also, probably, their business model and organizational processes, software vendors need to analyse if what they want to achieve, is actually feasible for them in terms of technology, processes and business.

This paper presents an approach for a technical feasibility analysis of a migration of an application to the cloud. The main aim of this analysis is twofold. On one hand, support the establishment of the most suitable migration tasks and on the other hand, provide an estimation of the required effort to implement these migration tasks with the final goal of supporting the decision making process prior to a modernization project.

II. MOTIVATION

Research literature and real industrial migration projects have documented several general procedures to estimate the cost and efforts required by a migration process, and therefore deciding on its feasibility.

Both analogy based estimation [2], that is, by comparing current migration project with other undergone migration projects and estimation given by experts' judgment [3] uses the knowledge in previous similar migration experiences, gained by experts to evaluate and estimate the complexity and efforts to undertake a new migration mission. Unfortunately, these approaches cannot be applied to migration project towards the Cloud, since the Cloud paradigm adoption is relatively recent, whereby the number of documented migration projects of legacy software to the Cloud is scarce [6, 7].

The most popular estimation approach is based on algorithmic models [4] that propose mathematical models to derive a quantitative estimation of migration costs based on identified costs factors. Although this approach also requires historical data in order to evaluate some parameters introduced by the mathematical models (i.e. weights in the model), its applicability is more generic than previous approaches, and therefore more suitable for a wider range of migration projects.

In order to estimate software development costs using metrics for software size measurement, some algorithmic methods based on Function Point Analysis (FPA) [5] have been proposed in literature. The FPA cost estimation is based on the analysis of software requirements.

FPA-based approaches can be more appropriate to estimate the complexity and provide effort/cost estimations (by historical data comparison) of migration tasks. In particular, FPA function points, in the context of a migration to Cloud project, can be mapped into migration tasks [6]. The systematic estimation of efforts required to migrate a legacy application into the Cloud has received less attention in the research community, notably because the migration to Cloud is a relative new concern. Up to our knowledge, only one work has proposed a systematic methodology for effort estimation of Cloud migration projects, namely Cloud Migration Point (CMP) [7], an adaptation of the FPA approach for software size estimation applied to the context of Cloud migration.

Complementing FPA-based approaches, there exist others based on software size estimation, including software complexity estimations. However, these methods can hardly be used on their own when wishing to estimate the size and complexity of the developments required migrating a legacy

 ISBN: 978-1-61208-367-4

application to the Cloud [8], because they do not offer enough information. Nonetheless, software size/complexity estimations on components of existing software systems can be used to classify the complexity of migrations tasks performed on these components, by comparing computed complexity metrics with historical data [9]. In particular, coupling metrics seems to help in the re-factoring of subsystems in an effective way to achieve the lower cost and high re-usability [10], which are factors to take into account when migrating to Cloud.

III. ARTIST APPROACH FOR TECHNICAL FEASIBILITY ASSESSMENT

A. *Mission and scope*

The ARTIST Technical Feasibility Tool (TFT) aims at supporting users on the early technical assessment of the migration of their applications to the Cloud. At this early stage (e.g. pre-migration phase in ARTIST Methodology [11]), the users need support to evaluate the feasibility of the migration, attending its technical aspects, since even for a very simple application, its migration to the Cloud may require non negligible efforts and concrete expertise to be accomplished. Moreover, the support for decision making 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.

B. *Functional description*

TFT works on Model Driven Engineering (MDE) representations (e.g. models) of the applications, particularly UML component models, offering to the users the following features:

- Visualization of components or features of the legacy application and the selection of those to be affected by the migration.
- Visualization of migration goals, which ultimately will drive the migration process. Migration goals can be obtained from the Cloud maturity assessment obtained through the ARTIST Maturity Assessment Tool (MAT) [12] or expressed by the user using the ARTIST Goal Modeling Editor [13].
- Identification of the required migration tasks on affected components. TFT suggests migration tasks per component. TFT allows users to confirm these tasks (optionally, TFT tries to select some tasks by default, but the user is able to override this selection anytime).Selection of weighted complexity estimations for every task type from expert judgment figures, initially taken from [7]. These figures provide task complexity weights estimated by experts based on accumulate experiences.
- Computation of complexity estimations for every component, calculating some metrics, in particular those metrics that estimate their maintainability.
- Computation of complexity estimations for a single task, as a function that considers both the complexity of the component affected by the task and task complexity itself
- Computation of effort estimations for a single task, as proportional to the computed task complexity, where the proportionality weight is given by expert judgment.
- Computation of global migration effort, by summing over individual migration task, for each migrated component.

C. *Technical approach*

Our implementation of TFT extends the CMP approach by automating some steps, using techniques explored by ARTIST such as Model Driven Reverse Engineering (MDRE), Software Metrics or Domain Specific Language (DSL)-based heuristics, notably to extract knowledge of the application, propose migration strategies and estimate the component complexity. CMP based computation of migration efforts is mostly conducted manually. On the contrary TFT is aiming to automate this process as much as possible, although leaving to the user the knowledge and control all over the entire process.

TFT approach to estimate the cost of the migration is based on the analysis of the migration requirements. Therefore, the specification of the overall objectives of the migration, that is, the migration goals, combined with the component-specific migration requirements and the preliminary Cloud target selection are inputs that will drive the TFT analysis. TFT leverages on high level model representations of the application, from which TFT elaborates a detailed breakdown analysis into components or features and creates a detailed structural breakdown of the migration process per legacy component. For such, TFT extracts legacy components from the high level model representations of the application, analyses their relationships and dependencies, determines their type (i.e. data sources, data entities, distributable services, controllers, views, etc.), estimates their complexity and maintainability (and possibly other metrics), and finally reports all these findings to the user in a component inventory view. TFT uses sources of domain-specific information, like expert judgment, to define heuristics that are used to infer the most appropriate migration strategies. These strategies are instantiated as migration tasks, for each component selected for migration, aiming at fulfilling the overall migration goals and the specific component migration requirements, addressing the Cloud target selection as well. TFT encodes these heuristics, used for task suggestion, in rules defined with a concrete domain specific language (DSL), in particular, tha JBoss Drools [14] DSL and engine is used., This approach avoids hardcoding expert judgment on TFT code implementation, which provides greater flexibility to extend the TFT knowledge in the future.

 ISBN: 978-1-61208-367-4

IV. TECHNICAL FEASIBILITY TOOL: DESIGN AND IMPLEMENTATION

A. General architecture

In Fig.1 the general architecture of the Technical feasibility Tool is depicted, and explained in section B.

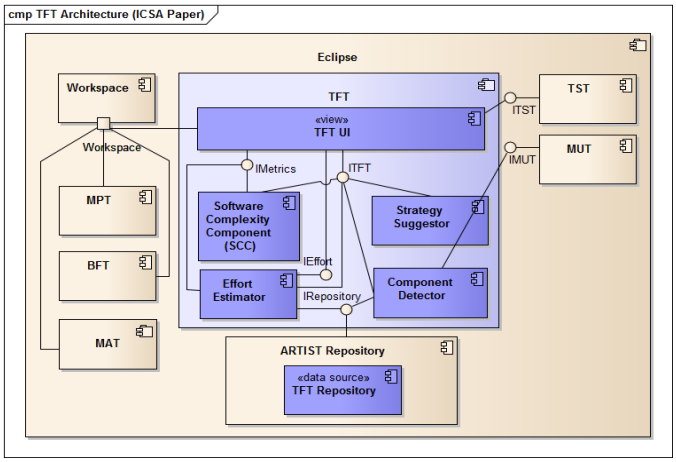

Figure 1. TFT general architecture

B. TFT components in detail

TFT consists of a set of Eclipse views and other widgets and wizards, a set of backend components and a set of external dependencies with other ARTIST components and tools, accessed through well-defined interfaces.

TFT UI complements the ARTIST Eclipse perspective with its collection of views on which the functionality of TFT is offered to the user:

- Navigator view: to browse and select existing legacy application projects
- Modelling view: to browse and annotate platform specific/platform independent models (PSM/PIM) component views provided by the ARTIST Model Understanding Tool (MUT) [15].
- Annotation View, provided by the ARTIST Target Specification Tool (TST) [15], which collects existing migration goals/requirements and provides support to annotate the existing legacy models in order to express additional migration goals.
- Inventory of components View: this TFT view collects the components from the component model and suggests migration strategies for each of them. The estimation of efforts for these migration strategies are also calculated and are shown to the user in a range of low, average and high for each migration strategy. The view allows modifying the migration strategies that affect them from a list of compatible strategies depending on the components' properties. This view also allows the user to select/deselect components to be considered to be migrated or not.
- Migration Goals View: allows user to browse and enable/disable the migration goals provided by MAT.
- Metrics view: this view allows selecting the metrics to be calculated for a selected component and displays the metrics figures.
- Effort estimation report view: this view reports the estimated effort for the overall migration project and individual migration tasks.

The TFT-UI makes use of these views, which are heavily dependent on RCP components such as Standard Widget Toolkit and JFace. Eclipse Workbench components are also used to make contributions to the Eclipse UI itself. TFT contributes to context menus of files with "uml" and "di" extensions and Papyrus [16] containers, with actions to open the Inventory View, and to context menus of files with XML extension to open the Migration Goals View. The TFT plugin also adds a listener to the opened component diagram files which listens the changes done to the file via using EMF/UML2 [17] or Papyrus editors.

TFT relies on several backend components to provide business logic support to TFT-UI.

- Components Detection component**:** It analyzes high level EMF Ecore UML2 PSM/PIM component models of the selected legacy application. The component uses EMF-Query to filter and EMF-Core and UML2 to analyse and modify the input model.
- Software Complexity component which computes a set of metrics on selected components. This component is explained in detail in the next section.
- Migration Strategy Suggestion component: It is responsible for analysing the components of the non-cloud compatible application and the relationships between them and suggesting certain migration strategies for each component to assist the user in the pre-migration process. Strategy suggestion process relies on a set of Drools rule defined in a DSL-based rule language which is interpreted by JBoss Drools. The strategy suggestion process is handled by the rule engine which is implemented using JBoss Drools
- Effort Estimation component: This component estimates the effort required to accomplish each required migration strategy suggested. The effort calculation is based on the migration strategy complexity and the complexity of the affected component(s). Strategy complexity is calculated using historical data and the expert knowledge encoded in the DSL based rules. Component complexity is provided by the Software Complexity Component. The final effort metric values are also based on expert knowledge combined with the complexity metrics.

 ISBN: 978-1-61208-367-4

- TFT Repository**:** This component stores historical data and heuristics required to estimate efforts.

1) Software Complexity Component

In order to evaluate the effort required to perform a migration task, TFT analyses several parameters as explained above in the paper. One of these parameters is the complexity related to the legacy software.

The estimation of the complexity of the legacy software is performed, by the Software Complexity Component (SCC). It provides information about how complex the legacy software is in terms of easiness to evolve it to the Cloud paradigm. This information is provided by means of software complexity metrics.

Software complexity has been defined and calculated in a vast variety of ways in the last years. Upon closer examination, these are some several commonly used metrics:

- McCabe Cyclomatic Complexity (v(G)) [18]
- Weighted Methods per Class (WMC) [19]
- Afferent Coupling (Ca) [20]
- Efferent Coupling (Ce) [20]
- Instability (I= Ce / (Ca + Ce)) [21]
- Number of Interfaces [21]

The correlation of these metrics is of highly importance, as a variation in one of them has an impact on the others. Literature has studied this correlation mainly for maintainability concerns which is defined by IEEE standard glossary of Software Engineering [22] as "the ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment".

The Compound MEMOOD method presented in [23], based on the MEMOOD model [24], creates a maintainability model based on the creation of 4 models: 1) Modifiability, 2) Understandability, 3) Scalability, 4) Level of complexity. Each of these models is based on metrics extracted from the source code and the class diagrams.

SCC uses the models cited beforehand in order to calculate the software maintainability index, the metric that ARTIST will use to measure the complexity of the legacy code.

These models use several metrics to calculate maintainability as the way to calculate the complexity. In the context of ARTIST project where the feasibility for a migration to cloud is being evaluated, the maintainability metric (as defined by IEEE) for calculating the software complexity will be used:

$$\textbf{Maintenance} = 2.399 + 0.493 \times Modifiability + 0.474 \times Understability + 0.524 \times Scalability + 0.507*LOC$$

$$\textbf{Modifiability} = 0.629 + 0.471 \times NC - 0.173 \times NGen - 0.616 \times NAggH - 0.696 \times NGenH + 0.396 \times MaxDIT$$

$$\textbf{Understability} = 1.66 + 0.256 \times NC - 0.394 \times NGenH$$

$$\textbf{Scalability} = 0.182 \times 0.99 \times AC + 0.100 \times EC + 0.097 \times ND - 0.036 \times PC + 0.068 \times DMS$$

$$\textbf{LOC} = 0.269 + 0.008 \times Coupling + 0.181 \times cohesion + 0.119 \times CC + 0.084 \times ILCC$$

The required metrics to perform these models are described in [25].

The aforementioned models have been predicted using data from several sources [26] using the multivariate linear model. However the correctness and fine-tuning of the formulas have to be updated to the context of ARTIST use cases.

There are several tools available in the Open Source community that offers some of the functionalities required by SCC. A first criterion to select the list of potential candidates to be re-used has been their availability as Eclipse plugin (as the basis technology of TFT and the majority of ARTIST tools), support to Java and C# and finally the availability of the source code. Following these criteria, three existing plug-ins where analyzed in detail, Metrics [27], Sonar [28] and CodeProAnlytix [29].

After a deep analysis of these tools, all of them have been discarded as they do not accomplish the requirements for the ARTIST project, rejecting also a possible adaptation of them for platform compatibility reasons.

The current SCC prototype architecture is a java API that explores source files and UML models to generate several metrics of a specific project. It comprises three sub-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 testing purposes. It generates several use cases that test the functionality of the SCC generating console logs and XML files with the results.

C. TFT validation

The first validation of all the components of the TFT has been performed executing in parallel:

1. The TFT comprising the TFT-UI, component detection component, strategy suggestion component, effort estimation component and TFT repository (see Fig. 3)
2. The Software Complexity Component, which calculates the maintainability index and other required metrics per component. (see Fig 2)

The component models of the Java version of the Petstore [30] application and two ARTIST use cases, Line of Business (LoB) [31] and Distant Early Warning System

(DEWS) [31] were used as sample inputs for TFT and SCC. Petstore is a multi-tier J2EE application, a B2C Web portal that displays a Pet catalog and support basic commerce. LoB is a .NET solution over Microsoft Sharepoint [32] for collaborative business process modeling. DEWS offers a complex SOA-based system (including desktop end-user command and control UIs) enabling the early detection and warning broadcasting of tsunami threats.

The component model of Petstore and DEWS were obtained using semi-automatic MDRE techniques, but the component model of LoB was created by hand. The MDRE process followed to obtain these models was as follows. Using Modisco [33], we obtained PSMs from the legacy code. These models were abstracted to a PIM level using a search-based model exploration approach [34], using either ATL [35] query and INC-Querying [36] techniques, combined with UML profiling [37] and slicing methods [38]. A further ATL M2M transformation generated a UML component model from the UML stereotyped classes existing in the PIM, aggregating similarly stereotyped classes within the same containment (i.e. package) to constitute components.

Two sample MAT reports were used (one for each platform) as the second input of TFT. TFT was fed with the MAT report and the component model of the legacy application in order to identify suggested migration strategies for each component of the application and compute the effort estimations for these strategies. TFT triggered its expert knowledge base (encoded as a set of rules) to suggest and select migration strategies for each component located in the input model. The migration complexity reported by TFT is the average of the complexity of selected strategies (information encoded in the TFT expert knowledge base as well). The estimated migration efforts are computed by TFT following a similar FP analysis conducted in [7] as the sum of efforts computed for each strategy selected for each component.

TFT was successful to deliver meaningful results in both migration suggestions and effort computations. In order to improve the quality of the suggestions, a deeper analysis on the components and its complexity metrics is required which is achievable by creating more complex rule definitions. The migration effort computation may be enhanced by increasing the number of evaluated applications thus enlarging the historical data.

In Fig. 2 and 3 the results for DEWS use case are shown:

```
****** Maintenance
Component: org.aspencloud.widgets Maintenance: 2.6357682
Component: org.aspencloud.widgets.cdatepicker Maintenance:
2.652078
Component: org.aspencloud.widgets.cnumpad Maintenance:
2.6818948
Component: org.aspencloud.widgets.snippets Maintenance:
2.7467294
Component: org.dews_online.ccui Maintenance: 2.5707283
Component: org.dews_online.ccui.control.jobs Maintenance:
2.590284
Component: org.dews_online.ccui.splashHandlers Maintenance:
2.6118982
```

Figure 2. SCC console log for DEWS (Maintainability metric)

Component Name	Stereotypes	Strategy Suggestion	Complexity	Estimated Effort
org.dews_online.cc				
MonitoringPers	Perspective ,SWT	Migrate to GWT	AVERAGE	7
ForecastingPers	Perspective ,SWT	Migrate to GWT	AVERAGE	7
MessageCompo	Perspective ,SWT	Migrate to GWT	AVERAGE	7
MessageDissem	Perspective ,SWT	Migrate to GWT	AVERAGE	7
ADewsBusiness	Subject ,JMS	Migrate to TQA	HIGH	7
SensorNetworkI	Subject ,JMS	Migrate to TQA	HIGH	7
SensorTypeTim	Subject ,JMS	Migrate to TQA	HIGH	7
DisseminationS	Subject ,JMS	Migrate to TQA	HIGH	7
DataModelCon	DataModel ,JMS	Migrate to TQA ,M...	HIGH	13

Figure 3 TFT Inventory View showing migration suggestions and efforts for DEWS

V. CONCLUSIONS

This paper presents a systematic approach that enables an early estimation of the complexity and the efforts required for the migration of existing applications to a Cloud provider. This approach combines traditional FPA techniques for migration task decomposition and effort estimation with others such as a) model-driven reverse engineering and model comprehension techniques to capture information about application components, b) expert judgment (for task suggestion and complexity estimation) implemented as a knowledge base of domain specific heuristics and c) complexity estimation (i.e. software maintainability) using an empirical combination of computable metrics. A prototypical implementation of this approach, available as an Eclipse plugin, has been described. Preliminary evaluation of the approach and tooling support has been conducted in an early evaluation of some case studies. This have enabled us to increase the TFT knowledge base of rules suggesting migration tasks and estimating their complexity, relying on the migration experiences gained through these cases. Nonetheless, the lack of reported experiences about migrating to Cloud has constrained our knowledge base to the expert judgment acquired in these few experiments and the effort figures reported on [6]. Nonetheless, the TFT decoupling between its knowledge base and its implementations eases the extension of the knowledge base as soon as new insights are gathered in other validation experiments. Foreseen future work, in the short term, includes: a) the integration of computed SCC metrics, in the computation of migration task efforts using empirical formulas that combines component maintainability with task complexity, b) the extension of TFT knowledge base to incorporate additional expert judgment heuristics to suggest additional Cloud optimization patterns, c) adjustment of the TFT effort figures collecting experimental data from ARTIST migration case studies.

VI. ACKNOWLEDGMENT

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.

 ISBN: 978-1-61208-367-4

VII. REFERENCES

[1] ARTIST project "Advance software based service provisioning and migration of legacy Software", http://www.artist-project.eu/ [retrieved: July, 2014].

[2] M. Shepperd,C. Schofield, "Estimating software project effort using analogies", IEEE Transactions on Software Engineering, vol. 23, no. 11, Nov. 1997, pp. 736-743.

[3] M. Jorgensen, "A review of studies on expert estimation of software development effort", Journal of Systems and Software, vol. 70, no. 1-2, 2004, pp. 37-60.

[4] M. Jorgensen and M. Shepperd, "A systematic review of software development cost estimation studies", IEEE Transactions on Software Engineering, vol. 33, no. 1, January 2007. pp. 33-53.

[5] A. Albrecht, J. Gaffney, "Software function, source lines of code, and development effort prediction: A software science validation", IEEE Transactions on Software Engineering, vol. 9, 1983, pp. 639-648.

[6] V. Tran, JW. Keung, A. Liu, A. Fekete. "Application Migration to Cloud: A Taxonomy of Critical Factors". SECLOUD 2011 Software Engineering For Cloud Computing Workshop, 2011, pp. 22-28.

[7] V. Tran, JW. Keung, A. Liu, A. Fekete ."Size Estimation of Cloud Migration Projects with Cloud Migration Point (CMP)", Proceedings of International Symposium on Empirical Software Engineering and Measurement (ESEM'11), Banff, Canada, September 2011, pp. 265-274.

[8] B. Touesnard, "Software Cost Estimation: SLOC-based Models and the Function Points Model". Version 1.1, 2004.

[9] H. Najadat, I., Alsmadi,Y., Shboul. "Predicting Software Projects Cost Estimation Based on Mining Historical Data", ISRN Software Engineering Volume 2012.

[10] H.Ramakrishnan, "Analysis of complexity and coupling metrics of subsystems in large scale software systems", M. S Thesis 2006.

[11] ARTIST Consortium "D6.2.1 ARTIST Methodology" http://www.artist-project.eu/sites/default/files/D6.2.1_ARTISTMethodology_M12_30092013.pdf [retrieved: July, 2014].

[12] ARTIST Consortium "Business and Technical Modernization assessment tool M12", http://www.artist-project.eu/sites/default/files/D5.2.1Businessand Technical Modernizationassessmenttool_M12_30092013.pdf, [retrieved: July, 2014].

[13] ARTIST Consortium "Methodology and Environment for evaluating migration success", http://www.artist-project.eu/sites/default/files/D11.3.1 Methodology and Environment for evaluating migration success M8_31052013.pdf [retrieved: July, 2014].

[14] JBoss Drools, http://www.jboss.org/drools/ [retrieved: July, 2014].

[15] ARTIST consortium, "ARTIST Integrated Architecture M15.

[16] Papyrus, http://www.eclipse.org/papyrus/, [retrieved: July, 2014].

[17] EMF /UML2 http://www.eclipse.org/modeling/mdt/?project=uml2 [retrieved: July, 2014].

[18] McCabe (1976). "A Complexity Measure". IEEE Transactions on Software Engineering http://www.literateprogramming.com/mccabe.pdf, [retrieved: July, 2014].

[19] http://metrics.sourceforge.net/ [retrieved: July, 2014].

[20] R. Martin. "OO Design Quality Metrics: An Analysis of dependencies". Workshop Pragmatic and Theoretical Directions in Object-Oriented Software Metrics. http://www.cin.ufpe.br/~alt/mestrado/oodmetrc.pdf. [retrieved: July, 2014].

[21] Instability http://en.wikipedia.org/wiki/Software_package_metrics, [retrieved: July, 2014]

[22] IEEE (1990). "IEEE Std 610.12-1990 - IEEE Standard Glossary of Software Engineering Terminology".

[23] Ch. Gautam,, S. Kang, "Comparison and implementation of software maintenance models". International Journal of Engineering Research & Technology (IJERT), Vol. 1 Issue 6, August 2012.

[24] S.W.A. Rizvi, R.A. Khan, "Maintainability Estimation Model for Object Oriented Software in Design Phase (MEMOOD)". Journal of Computing. Volume 2, Issue 4, April 2010

[25] M. Genero, M. Patiani, C. Calero, (2005)"A Survey of Metrics for UML Class Diagrams" Journal of Object Technology http://www.jot.fm/issues/issue_2005_11/article1/article1.pdf [retrieved: July, 2014]

[26] S. Muthanna, K. Kontogiannis, K. Ponnambalam, B. Stacey, "A maintainability model for industrial software systems using design level metrics". Published in IEEE Proceedings Seventh Working Conference on Reverse Engineering, pp.248-256, 2000.

[27] http://sourceforge.net/projects/metrics2/ [retrieved: July, 2014].

[28] http://docs.codehaus.org/display/SONAR/Using+Sonar+in+Eclipse [retrieved: July, 2014].

[29] https://developers.google.com/java-dev-tools/codepro/doc/ [retrieved: July, 2014]

[30] Java PetStore http://www.mia-software.com/html/miaStudio/download/modisco/examples/javapetstore-2.0-ea5.zip [retrieved: July, 2014]

[31] ARTIST Consortium "Use cases definition and migration architecture" (2012) http://www.artist-project.eu/sites/default/files/D12.1 Use Cases definition and migration architecture_M12_01102013.pdf [retrieved: July, 2014]

[32] Microsoft Sharepoint: http://office.microsoft.com/en-001/sharepoint/ [retrieved: July, 2014]

[33] H. Brunelière, J. Cabot, G. Dupé, F. Madiot. MoDisco: a Model Driven Reverse Engineering Framework.Information and Software Technology 56, 8 ,2014, pp.1012-1032

[34] P. Baker , M. Harman , K. Steinhofel , A. Skaliotis, Search Based Approaches to Component Selection and Prioritization for the Next Release Problem, Proceedings of the 22nd IEEE International Conference on Software Maintenance, p.176-185, September 24-27,

[35] F., Jouault, and I. Kurtev,:On the Architectural Alignment of ATL and QVT. In: Proceedings of ACM Symposium on Applied Computing (SAC 06), Model Transformation Track. Dijon (Bourgogne, FRA), April 2006 [retrieved: July, 2014]

[36] INC-Query, http://www.eclipse.org/incquery/

[37] J. Cabot, C. Gómez. A simple yet useful approach to implementing UML Profiles in current CASE tools. In Workshop in Software Model Engineering, 2003.

[38] A., Blouin, B., Combemale, B., Baudry, O., Beaudoux. "Modeling Model Slicers". Model Driven Engineering Languages and Systems. Lecture Notes in Computer Science Volume 6981, 2011, pp 62-76

 ISBN: 978-1-61208-367-4