Quality Measurement Scenarios in Software Migration

Gaurav Pandey, Jan Jelschen, Dilshodbek Kuryazov, Andreas Winter
Carl von Ossietzky Universität, Oldenburg, Germany
{pandey,jelschen,kuryazov,winter}@se.uni-oldenburg.de

Abstract. Legacy systems are migrated to newer technology to keep them maintainable and to meet new requirements. To aid choosing between migration and redevelopment, a quality prognosis of the migrated software, compared with the legacy system is required. Moreover, as the driving forces behind a migration effort differ, migration tooling has to be tailored according to project-specific needs, to produce a migration result meeting significant quality criteria.

Available metrics may not all be applicable identically for both legacy and migrated systems, e.g. because of paradigm shifts during migration. To this end, this paper describes identifies three scenarios for utilizing quality measurement in a migration project.

1 Introduction

Migration, i. e. transferring the legacy systems into new environments and technologies without changing their functionality, is a key technique of software evolution [4]. It removes the cost and risk of developing a new system from scratch and allows to continue modernization of the system. However, it needs to be found out whether the conversion leads to a change in internal software quality. To decide between software migration and redevelopment, the quality measurement and comparison of legacy and migrated systems is required. Moreover, a migration project requires an especially tailored toolchain [3]. To choose the tools to carry out an automatic migration, assessment of the quality of migrated code is needed against the combination of involved tools.

The identification of project-specific quality criteria and corresponding metrics for quality comparison can be achieved with the advice from project experts. However in a language based migration, e.g. from COBOL to Java, there is a shift from procedural to object-oriented paradigm. This can lead to limiting the usability of a metric, as its validity and interpretation might not hold in both platforms. For example, the metrics calculating object-oriented properties like inheritance or encapsulation, can be used on migrated Java code but not on COBOL source code. To overcome this, it is required to have a strategy regarding utilization and comparison of metrics in migration. To this end, this paper identifies the quality measurement scenarios with suitable metrics, enabling the quality calculation in different situations. The next two sections explain the Q-MIG project and the measurement scenarios and are followed by a Conclusion.

2 Q-MIG Project

The Q-MIG-project (Quality-driven software MIGration)[1] is a joint venture of *pro et con Innovative Informatikanwendungen GmbH, Chemnitz* and *Carl von Ossietzky University's Software Engineering Group*. Q-MIG is aimed at advancing a toolchain for automated software migration [2]. To aid in deciding for or against a migration, selecting a migration strategy, and tailoring the toolchain and individual tools, the toolchain is to be complemented with a *quality control center* measuring, comparing, and predicting internal quality of software systems under migration.

The project aims at enabling quality-driven decisions on migration strategies and tooling [6]. To achieve this, the *Goal / Question / Metric* approach [1] is used. The goal is to measure and compare the quality of the software before and after migration, to enable migration decisions and toolchain selection. The questions are the quality criteria based on which the quality assessment and comparison needs to be carried out. The Q-MIG project considers internal quality attributes, i. e. focuses on quality criteria *maintainability* and *transferability* in terms of the ISO quality standard [5]. Moreover, expert advice is taken for selecting and identifying criteria relevant for software migrations. For example, maintainability-related metrics are important in a project that needs to keep on evolving, but not when the migrated project is meant to be an interim solution, until a redeveloped system can replace it. Then, to measure the quality criteria, metrics need to be identified. However, a metric that is valid for the legacy code might not be valid for the migrated code and vice versa. In order to identify the metrics for quality criteria calculation, the metrics are categorized as per the use case they can be utilized in. To achieve this, scenarios for quality comparison and toolchain component selection are defined in Section 3.

3 Measurement Scenarios

This section presents the quality measurement scenarios that utilize the quality metrics according to the properties measured and their applicability to the legacy and migrated platforms. While the first two scenarios facilitate the quality comparison between

[1] Q-MIG is funded by Central Innovation Program SME of the German Federal Ministry of Economics and Technology – BMWi (KF3182501KM3).

the legacy and the migrated systems, the third scenario is particularly useful for selecting components of the migration toolchain. While the Q-MIG project focuses on quality measurement of a COBOL to Java migration, the essence of the scenarios presented remains the same for other combinations of platforms.

Same Interpretation and Implementation: This scenario facilitates quality comparison of legacy code (COBOL) and migrated code (Java) to help in project planning. It is achieved by utilizing the quality metrics that are valid and have the same implementations and interpretations in both platforms, and hence allowing for direct quality comparison between the systems. For example, Lines of Code, measuring the size of the project, is calculated identically for COBOL and Java (In some cases Lines of Code can be platform specific requiring adaptations like Function Point Analysis). Similarly Number of GOTOs, Comments Percentage, Cyclomatic Complexity (Mc Cabe Metric) and Duplicates Percentage can be calculated for both languages in the same fashion.

Same Interpretation Different Implementation: In this scenario the metrics that have different implementations but same interpretation in legacy and target code are utilized for quality comparison. COBOL and Java codes are different in construct and the building blocks. So, certain metrics can have same interpretation but different ways of calculation in the platforms. For example, Cohesion is the degree of independence between the building blocks of a system. So, it can be calculated in the COBOL code considering procedures as building blocks, while in the migrated Java code they can be represented by classes. The two calculations can provide comparable metrics, hence enabling quality comparison. Similarly, other metrics can be utilized for quality comparison, that might not have exactly the same implementation for COBOL and Java. Some metrics conforming to this scenario are: Halstead's metrics (because it uses operators and operands, that differ among the languages), Average Complexity per Unit and Average Unit Size.

Target Specific Metrics: In this scenario, the metrics that are specific to target platform Java (and may not be applicable to COBOL legacy code) are utilized for toolchain selection and improvement. For example, the metric Depth of Inheritance can be calculated for Java, but not for COBOL (procedural languages have no inheritance). Also, value of the metric can change on changing components of migration toolchain or by additional reengineering steps. This allows to use the metrics to choose a suitable toolchain by analyzing how the quality of migrated software changes with respect to the chosen components.

But, in a one-to-one migration from COBOL to Java that introduces no restructuring, Depth of Inheritance metric value would not change with respect to the migration tools. This is because such migration will not introduce inheritance in the target code. However, the source code can be refactored before migration. And, an analysis of the metrics against the combination of refactoring tools allows the selection of the components of the refactoring toolchain.

This scenario allows the metrics relevant to Q-MIG project and applicable for Java code, to be utilized for selecting the migration and refactoring tools. Here, various object-oriented metrics are used like: Number of Classes representing level of abstraction in code. Also, Attribute Hiding Factor and Method Hiding Factor that calculate the percentages of hidden attributes and methods respectively, are related to modifiability. Moreover, Average Number of Methods per Class calculates complexity of the code. Also, the metrics that are applicable in previous two scenarios can be used here, as they are applicable to the migrated code. However, the reverse might not be true.

4 Conclusion

This paper identified three scenarios for measuring and comparing internal quality of software systems under migration, paired with applicable metrics. The scenarios stress the challenge of comparing quality measurements in the context of paradigm shifts, e. g. when migrating from procedural COBOL to object-oriented Java. They delimitate pre-/post-migration comparison to assess suitability of migrating, from comparing migration results using different toolchain configurations to improve the tools and tailor the toolchain to project-specific needs. Further steps in the project include the design and evaluation of a quality model by identifying relevant quality criteria, and making them measurable using appropriate metrics, with the scenarios providing an initial structure.

References

[1] V. R. Basili, G. Caldiera, and H. D. Rombach. The goal question metric approach. In *Encyclopedia of Software Engineering*. Wiley, 1994.

[2] C. Becker and U. Kaiser. Test der semantischen Äquivalenz von Translatoren am Beispiel von CoJaC. *Softwaretechnik-Trends*, 32(2), 2012.

[3] J. Borchers. Erfahrungen mit dem Einsatz einer Reengineering Factory in einem großen Umstellungsprojekt. *HMD*, 34(194):77–94, mar 1997.

[4] A. Fuhr, A. Winter, U. Erdmenger, T. Horn, U. Kaiser, V. Riediger, and W. Teppe. Model-Driven Software Migration - Process Model, Tool Support and Application. In A. D. Ionita, M. Litoiu, and G. Lewis, editors, *Migrating Legacy Applications: Challenges in Service Oriented Architure and Cloud Computing Environments*. IGI Global, Hershey, PA, USA, 2012.

[5] ISO/IEC. ISO/IEC 25010 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models. Technical report, 2010.

[6] J. Jelschen, G. Pandey, and A. Winter. Towards quality-driven software migration. In *Proceedings of the 1st Collaborative Workshop on Evolution and Maintenance of Long-Living Systems*, 2014.