Measuring Software Maintainability

dr. Adam Vanya

INTRODUCTION

In our previous blog post titled “The Effects of Software Maintainability”, we reviewed what maintainability can potentially influence. To manage the actual consequences, we must be able to control maintainability itself. This requires decision making and the execution of all the supporting actions. The decision making process applied can either be intuition- or fact-based. In this blog post, we will learn about how to measure the maintainability of software systems to extract facts for reliable and well informed decisions.

MAINTAINABILITY RELATED MEASUREMENTS

Our feelings and intuitions can be useful under certain circumstances. For instance, when we need to make decisions quickly and without the possibility to execute objective measurements. Our intuitions may turn out to be accurate or incorrect but they provide us at least a chance to achieve what we want. Taking a chance is better than sure failure.

Under some other circumstances, following the intuition-based decision making approach is not recommended. When, for instance, the success of your company and your personal career is at stake, then it is wise to base your important decisions on facts. Fact-based decisions are better substantiated and therefore, they turn out to be right more often. Good decisions are the stepping stones towards reaching your goals set. Revealing the needed facts, however, takes time and it is usually done by executing objective measurements.

Measuring software maintainability is essential to control maintainability based on facts. The measurements can be done directly and indirectly. Direct measurements target the process of maintenance and collect facts about how efficiently developers execute the modifications required. The question on how much time it takes to fix a bug or introduce an enhancement is answered by direct measurements of software maintainability. These direct measurements have the advantage of being able to provide us facts about software maintainability itself. The data collected this way can be used to decide whether intervention is needed or not. However, they do not enable us to decide about which actions should be executed to implement the intervention required. This is the point where indirect measurements have an added value.

Indirect measurements of software maintainability involve measuring those factors that affect software maintainability. As described in our blog post titled “Factors Influencing Software Maintenance”, these factors include primarily the state of the source code and some properties of the developers. Indirect

measurements complement direct measurements because they can signal what exactly should be improved in order to enhance maintainability.

Many properties of the source code can be measured. These include, besides others, the size and complexity of the programming units (like methods and functions), the amount of couplings between the components of the system and the ratio of redundant code which emerge as a consequence of code duplications. As for developers, we may be interested in revealing how much they know about software maintainability, how many things they can remember at the same time, how well they have mastered the technology used to implement the software system, how much motivation they have or whether they have all the resources (e.g. time) to implement maintainable code.

Of course, indirect measurements should carefully be executed as not all source code or developer properties have an equal influence on software maintainability. For instance, on which day of the week a method was modified is likely to have less effect on maintainability than the number of decision points implemented in such a method. Similarly, the eye or skin color of a developer is also less relevant than the amount of knowledge such developers have on maintainability. Regarding the maintainability-related properties, it may be tempting to measure as many of them as possible, but it is sufficient to measure some key indicators only. This ensures getting the most value for a limited amount of measurement costs.

The results of properly executed indirect software maintainability measurements can be used to predict how efficiently developers will be able to maintain the software under investigation. To this end, the measurement results of a software system are typically interpreted within the context of a benchmark that represents the market. The resulting scores are then aggregated to form a single maintainability score. The higher this final maintainability score is, the more easily developers are able to maintain the source code.

How exactly one can get to a maintainability score is described by measurement models. Such a model is described, for instance, in the ISO/IEC 25010:2011 standard (see [1]). Although this model serves as a widely used reference model for decomposing the concept of maintainability, the method it proposes to execute software related measurements is often not applicable in practice. Other, more practically applicable models include the maintainability index, the SQUALE model described in [2], the maintainability measurement models of FrontEndART (see [3]) and the Software Improvement Group (SIG) (see [4]).

Measuring a single snapshot of a software system already helps enormously to understand and predict software maintainability of that version. Based on the measurement results, one can decide to take actions if the measured level of maintainability is not sufficient. To make such decisions even better informed, trends should be considered as well. Depending on how the measured maintainability of two software systems changed over time, the required interventions may differ even if the last versions of those systems have the same level of measured maintainability. Measurement trends help us understand if we are getting closer or further away from our maintainability targets.

It has to be pointed out that all the models mentioned measure source code only and they do not consider developer properties in any way. However, measuring the knowledge, skill-levels, motivation and needs of the developers would be equally important to fully understand the extent of their impact maintainability. This is an area where we believe more research should be carried out.

SUMMARY AND CONCLUSIONS

One way or another, software maintainability is affecting you. It can have a significant impact on your career if you are working for a software development company and it is a major factor to consider when you invest in such companies or buy custom software. Therefore, we believe maintainability has to be understood and explicitly be controlled to reach your goals.

To help you make informed and fact-based decisions about software maintainability, we discussed what, how and why should be measured. We noted that next to the source code of software systems, also the properties and circumstances of the developers have to be investigated.

The next blog post in our series on software maintainability will explain how we can use the measurement results to decide about what the acceptable level of maintainability is. With these blog posts, we hope to increase awareness about software maintainability, help you control this concept and start constructive discussions about it.

In case there are certain things described in this blog post you do not or not completely agree with or if you think it should be extended in some ways, please feel free to contact us and tell us about your idea: adam.vanya@frontendart.com.

REFERENCES

[1] ISO/IEC 25010 (2011). ISO/IEC 25010:2011, Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models

[2] Mordal-Manet, K., Balmas, F., Denier, S., Ducasse, S., Wertz, H., Laval, J., …, Vaillergues, P. (2009). The squale model — A practice-based industrial quality model. 2009 IEEE International Conference on Software Maintenance, 531–534.

[3] FrontEndART Ltd. Quality model used in QualityGate.
Retrieved from https://cloud.quality-gate.com/documentation/qualitymodels.html

[4] Heitlager, I., Kuipers, T., & Visser, J. (2007). A Practical Model for Measuring Maintainability. 6th International Conference on the Quality of Information and Communications Technology (QUATIC 2007), 30–39.