How Much Does Unused Code Matter for Maintenance?

Sebastian Eder, Maximilian Junker, Benedikt Hauptmann, Elmar Juergens, 
Rudolf Vaas, Karl-Heinz Prommer

Institut für Informatik
Technische Universität München
Boltzmannstr. 3
85748 Garching b. München
{eders, junkerm, hauptmab}@in.tum.de
juergens@cqse.eu
{rvaas,hprommer}@munichre.com

Abstract: Software systems contain unnecessary code. Its maintenance causes unnecessary costs. We present tool-support that employs dynamic analysis of deployed software to detect unused code as an approximation of unnecessary code, and static analysis to reveal its changes during maintenance. We present a case study on maintenance of unused code in an industrial software system over the course of two years. It quantifies the amount of code that is unused, the amount of maintenance activity that went into it and makes the potential benefit of tool support explicit, which informs maintainers that are about to modify unused code.

1 Introduction

Many software systems contain unnecessary functionality. In [Joh], Johnson reports that 45% of the features in the analyzed systems were never used. Our own study on the usage of an industrial business information system [JFH+11] showed that 28% of its features were never used.

Maintenance of unnecessary features is a waste of development effort. To avoid such waste, maintainers must know which code is still used and useful, and which is not. Unfortunately, such information is often not available to software maintainers.

Problem: Real-world software contains unnecessary code. Its maintenance is a waste of development resources. Unfortunately, we lack tool support to identify unnecessary code and empirical data on the magnitude of its impact on maintenance effort.

Contribution: We contribute a case study that analyzes the usage of an industrial business information system of the reinsurance company Munich Re over the period of over 2 years. The study quantifies maintenance effort in unused code and shows the potential benefits of the tool support we propose.

Remarks: A complete version of the paper, the study, and a presentation of our tool support can be found in [EJJ+12].
2 Study and Results

We conducted the study on the level of methods in the sense of object oriented programming. The systems contain 25,390 methods. Of these, 6,028 were modified with a total of 9,987 individual modifications. This means that considerable maintenance effort took place during the analysis period.

RQ1: How much code is unused in industrial systems? We found that 25% of all methods were never used during the complete period.

RQ2: How much maintenance is done in unused code? We first compared the degree of maintenance (i.e. percentage of maintained methods) between used and unused methods. We found that 40.7% of the used methods were maintained, but only 8.3% of the unused methods. That means, unused methods were maintained less intensively than used methods. The unused methods account for 7.6% of the total number of modifications.

RQ3: How much maintenance in unused code is unnecessary? We reviewed examples of unused maintenance with the developers. By inspecting the affected code and researching the reason why it is not used, we found that in 33% of the cases, the unused code was indeed unnecessary. In another 15% of the cases the code in question was no longer existent as it was either deleted or moved. That means that in nearly every second case unused methods were either unnecessary or potentially deleted from the system.

RQ4: Do maintainers perceive knowledge of unused code useful for maintenance tasks? We encountered great interest in the analysis results, especially in the cases in which unused methods were maintained. Often, the developers were surprised that the respective method was not used.

3 Summary

We believe that our analysis would show a greater amount of unnecessary maintenance for projects with a different structure of the maintaining team, since the maintaining team knew the system very well. Due to the results, we are optimistic that this analysis helps directing maintenance efforts more effectively.

References

