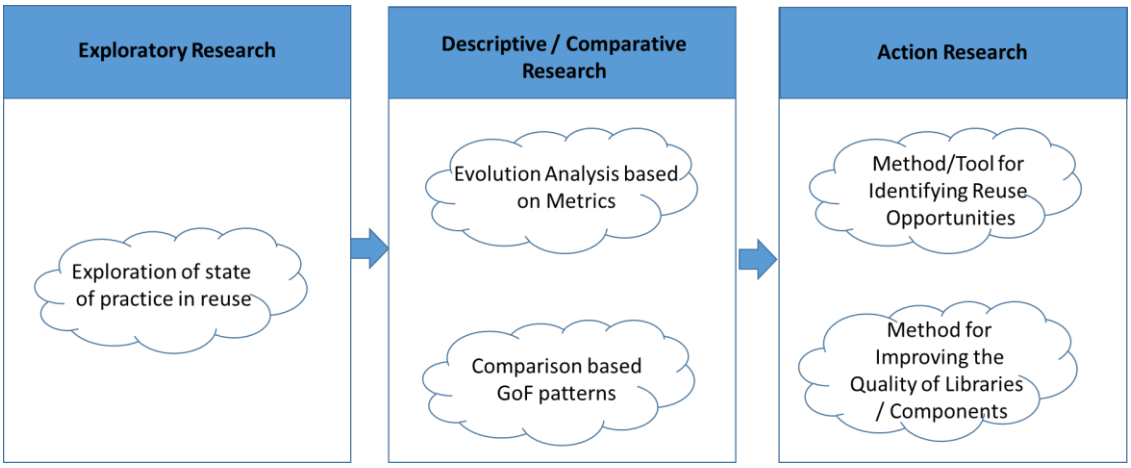
Systematic software reuse through open source libraries & components

Apostolos Ampatzoglou[1], Ioannis Stamelos[1], Alexander Chatzigeorgiou[2], Theodore Chaikalis[2], Ignatios Deligiannis[3], Panagiotis Sfetsos[3]



Project Overview

State-of-practice: Software development based on third-party libraries is becoming increasingly popular in recent years. Nowadays, the plethora of open-source libraries that are freely available to developers, offer great reuse opportunities, with relatively low cost. However, the reuse process is in many cases rather ad-hoc. In order for software reuse to advance from an opportunistic activity to a well-defined, systematic process, the reuse phenomenon should be empirically studied in a real-world setting. Thus, as a review of current state-of-practice, we assessed the: (a) extent to which software functionality is built from scratch or reused, (b) strategy and intensity of reuse activities in OSS development, (c) effect of reuse activities on design quality, (d) modification of reuse decisions from a chronological viewpoint, and (e) effect of these modifications on software design quality. To achieve these goals, we performed a large-scale embedded multi-case study on 1,111 Java projects, extracted from OSS repositories. The results of the case study provided a valuable insight on reuse processes in OSS development that can be exploited by both researchers and practitioners [1 and 2].

Quality of software libraries: Software libraries are expected to be well-designed, since they have to adhere to specific principles so as to accommodate the needs of multiple clients in a robust and stable way. Considering that most libraries are continuously upgraded, we investigated the evolution of their quality over time. In particular, we performed a case study to assess whether quality, in terms of three software metrics (CBO, LCOM, and WMC), exhibits clear trends during the history of twenty analyzed libraries. The findings indicate that the examined software libraries can be considered as stable software projects in terms of quality, in the sense that in contrast to the general belief about software aging, their quality does not degrade over time [3].

[1] Department of Informatics, Aristotle University, Thessaloniki, Greece, apamp@csd.auth.gr, stamelos@csd.auth.gr
[2] Department of Applied Informatics, University of Macedonia, Thessaloniki, Greece, achat@uom.gr, chaikalis@uom.gr
[3] Department of Information Technology Technological Education Institute, Thessaloniki, Greece, ignatios@it.teithe.gr, sfetsos@it.teithe.gr

In addition to software metrics, the existence of GoF design pattern instances is often regarded as an indication of elaborate software design, since patterns have been reported in many studies as techniques that improve software quality properties. Driven by the widespread belief that software libraries excel in terms of design quality compared to standalone applications, we investigated whether this claim is confirmed and if the improved quality can be attributed to the use of patterns. The results of the study suggested that, some of the quality properties are improved in library software although no significant difference in the use of patterns have been observed. Moreover, there is an important number of patterns that appears to be correlated to metrics [4].

Identification of reuse opportunities: Component-Based Software Engineering focuses on the development of reusable components to enable their reuse in more systems, rather than only to the ones for which they have been implemented in the first place (i.e. development for reuse) and the development of new systems with reusable components (i.e. development with reuse). To this end, we introduced a methodology for the extraction of candidate reusable components. The extracted components have been empirically evaluated through a case study, and are available for reuse through a webpage [5].

Quality Improvement Process: Refactoring aims to improve the design of existing code to cope with foreseen software architecture evolution. The selection of the optimum refactoring strategy can be a daunting task involving the identification of refactoring candidates, the determination of which refactorings to apply and the assessment of the refactoring impact on software product quality characteristics. As such, the benefits from refactorings are measured from the quality advancements achieved through the application of state of the art structural quality assessments on refactored code. Traditional, quality evaluation methods fall short in examining the financial implications of uncertainties imposed by the frequent updates/modifications and by the dynamics of the XP programming. To this end, we proposed the application of a simple Real Options Analysis technique, perceiving the selection of the optimum refactoring strategy as an option capable of generating value (cost minimization) upon adoption. To get an estimation of the expected cost that is needed to apply the considered refactorings and to the effect of applying them, in the cost of future adoptions we conducted a case study. The results of the case study suggested that every refactoring can be associated with different benefit levels during system extension [6].

References

[1] E. Constantinou, A. Ampatzoglou, and I. Stamelos, "Quantifying reuse in OSS: A large-scale empirical study", *International Journal of Open Source Software and Processes*, 2015.

[2] A. Zaimi, A. Ampatzoglou, N. Triantafyllidou, A. Chatzigeorgiou, A. Mavridis, T. Chaikalis, I. Deligiannis, P. Sfetsos, and I. Stamelos, "An Empirical Study on Reusing Third-Party Libraries in Open-Source Software Development", *7th Balkan Conference on Informatics*. ACM, 2015.

[3] T. Chaikalis, A. Chatzigeorgiou, A. Ampatzoglou, and I. Deligiannis, "Assessing the Evolution of Quality in Software Libraries", *7th Balkan Conference on Informatics*. ACM, 2015.

[4] P. Sfetsos, A. Ampatzoglou, A. Chatzigerorgiou, I. Deligiannis, and I. Stamelos, "A comparative study on the effectiveness of patterns in software libraries and standalone applications", *Proceedings of the 9th International Conference on the Quality of Information and Communications Technology (QUATIC),* IEEE Computer Society, pp. 145–150, 2014.

[5] A. Ampatzoglou, I. Stamelos, A. Gkortzis, and I. Deligiannis, "A methodology on extracting reusable software candidate components from open source games", *Proceeding of the 16th International Academic MindTrek Conference*, New York, NY, USA. ACM, pp. 93–100, 2012

[6] A. Mavridis, A. Ampatzoglou, I. Stamelos, P. Sfetsos, and I. Deligiannis, "Selecting Refactorings: An Option Based Approach", *8th International Conference on the Quality of Information and Communications Technology (QUATIC' 12)*, IEEE Computer Society, pp. 272–277, 2012.