Identifying Architectural Technical Debt in Android Applications through Automated Compliance Checking

Roberto Verdecchia

mail: *roberto.verdecchia@gssi.it*
supervisor: Patricia Lago
co-supervisor: Ivano Malavolta

Questions? Look for me around!

Overview

Mobile application business model is tightly coupled to users satisfaction. Promptly and efficiently releasing new versions to introduce new features, fix bugs, and adapt to users' needs is crucial. *Architectural technical debt (ATD)* hinders by definition evolvability. Among other techniques, compliance checking is used to undercover ATD [1]. Recently, effort was spent in standardizing Android architectural components to lower complexity of Android apps and provide a Android architectural guidelines [2]. We present a *novel approach*, based on (i) *guideline extraction*, (ii) *architecture reverse engineering* and (iii) *compliance checking*, for identifying ATD hotspots in Android apps.

IN FEW WORDS

what (project goals):

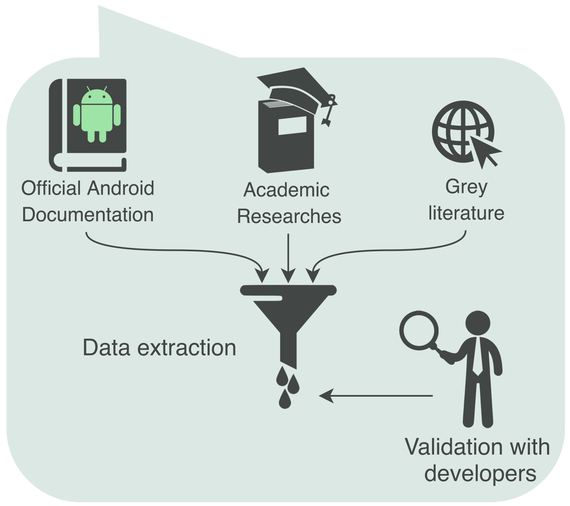
- ✓ Reusable *Android Reference Architecture*
- ✓ Automated process for architectural technical debt hotspot analysis of Android applications

how (technique):

- ➢ Manual extraction and formalization of Android architectural guidelines
- ➢ Automated *implemented architecture* reverse engineering
- ➢ Automated model-based compliance checking
- ➢ Quantitative assessment of compliance-violations to identify hotspot components

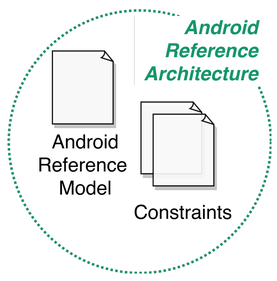
1 Guidelines extraction

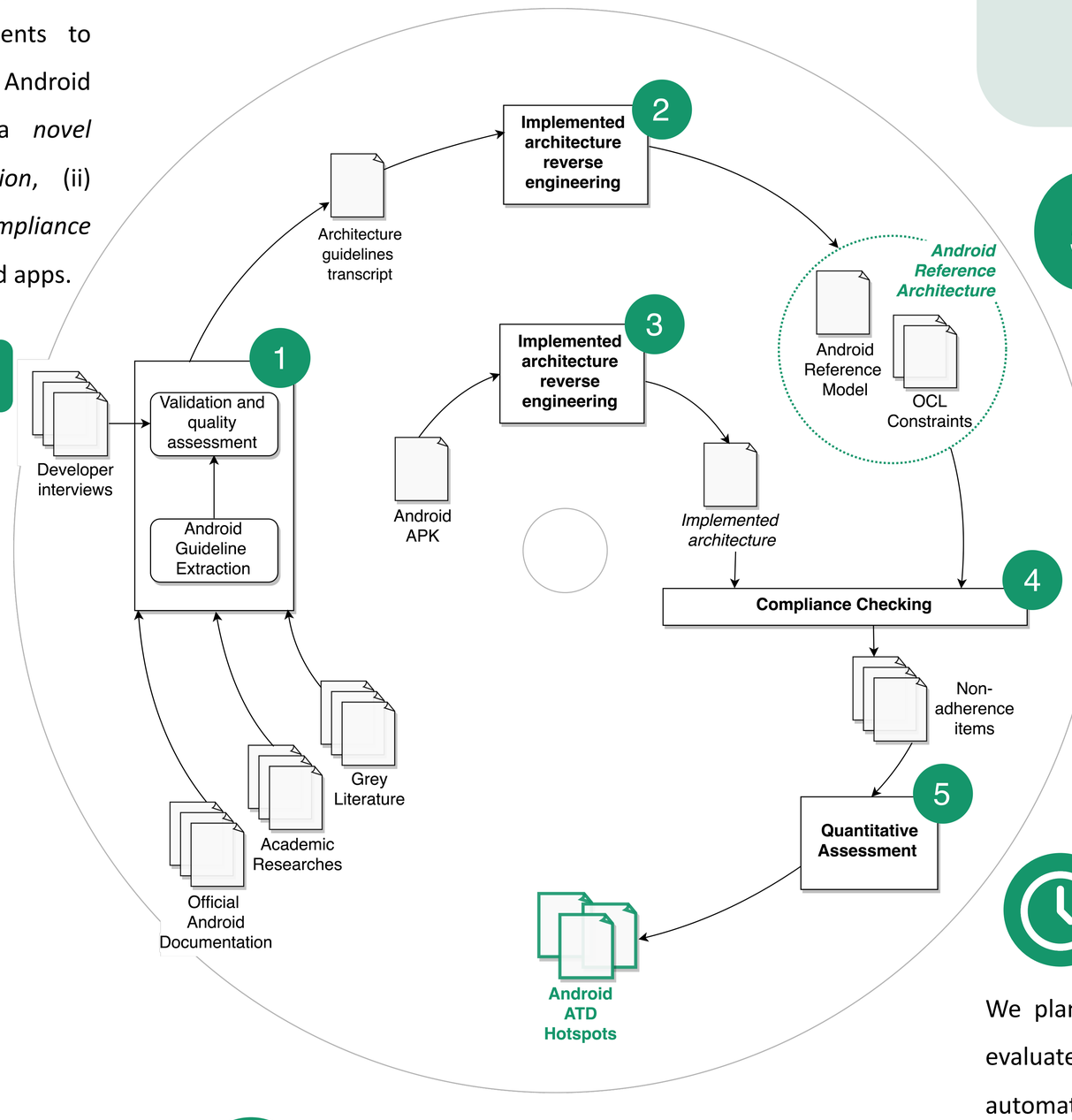
The first step consists in an *architectural guideline extraction from heterogeneous sources*. Guidelines should embed architectural rules designed to avoid incurring in potential ATD. The extracted guidelines are validated through developers interviews.



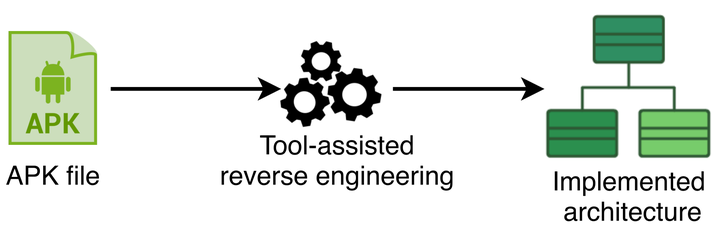
2 Reference architecture

This step consists in developing an ADL *Android reference model* conveying the architectural guidelines. Additionally, a *set of design constraints is established (*e.g. *expressed* through an Object Constraint Language). The combination of reference model and constraints is referred to as *Android reference architecture*.





3 Architecture reverse engineering

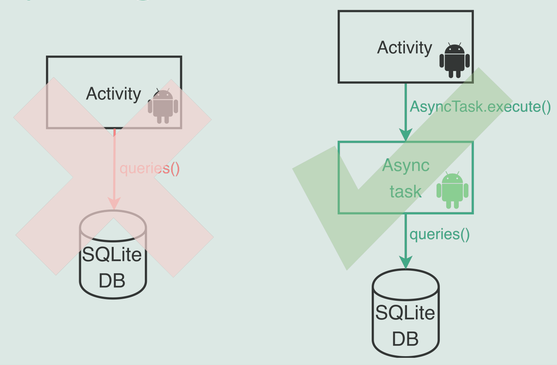


This step consists in the reverse engineering of the architecture of an implemented Android application through the analysis of its source code or APK. This process is carried out by extracting the most relevant Android architectural components and their implementation [3]. The reference architecture and the implemented one must adhere to the same metamodel or be linked by a suitable model-to-model transformation

4 Compliance checking

During this process, items of non-adherence of the *implemented architecture* w.r.t. the *Android reference architecture* are identified. The items are stored for a subsequent analysis carried out in Step 5. Due to its complexity this step has to be carried out (semi) automatically through a model comparison tool.

Example of guideline violation



5 Quantitative assessment

Once the set of non-adherence items is computed, it is possible to analyze the gathered data to identify which architectural elements of the implemented architecture violate the highest number of Android architectural guidelines. Distinct violation types can even be associated to specific weights to support a more involved prioritization process. The final architectural elements identified through this process are referred to as *Android ATD hotspots.*

Future work and outlook

We plan to fully automate the process and extensively evaluate it on large dataset of apps. In addition, the fully automated process enables us to carry out evolutionary studies of ATD in Android, through which a higher precision of the approach could be achieved.

References

[1] R.Verdecchia, I.Malavolta, and P.Lago. *Architectural Technical Debt Identification: The Research Landscape*. In TechDebt 2018.

[2] *Android and Architecture*. Android Developers Blog. https://androiddevelopers.google.blog.com/2017/05/android-and-architecture.html

[3] H.Bagheri, J.Garcia, A.Sadeghi, S.Malek, and N.Medvidovic. 2016. *Software architectural principles in contemporary mobile software: from conception to practice*. JSS 119 (2016), 31–44.

WANT MORE INFO?

Download the abstract and more at
https://roberto.verdecchia.github.io/

GRAN SASSO SCIENCE INSTITUTE
SCHOOL OF ADVANCED STUDIES
Scuola Universitaria Superiore

VRIJE UNIVERSITEIT AMSTERDAM

MOBILESoft2018
5TH IEEE/ACM INTERNATIONAL CONFERENCE ON MOBILE SOFTWARE ENGINEERING AND SYSTEMS