Detection of Anti-patterns in Mobile Applications
Geoffrey Hecht, Laurence Duchien, Naouel Moha, Romain Rouvoy

To cite this version:
Geoffrey Hecht, Laurence Duchien, Naouel Moha, Romain Rouvoy. Detection of Anti-patterns in Mobile Applications. COMPARCH 2014, Jun 2014, Lille, France. <hal-01029994>

HAL Id: hal-01029994
https://hal.inria.fr/hal-01029994
Submitted on 22 Jul 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
**Motivations**

Lack of studies on Android Anti-patterns:
- Influence of the framework on common anti-patterns,
- Android specific anti-patterns,
- Patterns related to applications categories,
- Correlation with application ratings,
- Evolution along updates?

**Anti-Patterns**

- Bad solutions to solve problems
- Hard to maintain and update
- Source of bugs

"God class" Anti-pattern:
- Low cohesion, numerous lines of code,
- A high coupling

**Application Analysis**

Analysis on **Android package**

- Analyze of **binary code** with Soot and dexpler submodule
- Metrics are extracted from Soot model to build a new one
- This model is then stored in a **graph database**

**Metrics**

Main metrics:
- Lines of Code
- Cohesion of a class
- Coupling between classes
- Cyclomatic complexity
- Depth of Inheritance

Anti-patterns = Combination of metrics
"God class" = low cohesion, numerous lines of code, a high coupling

**Anti-Patterns Detection**

Study on **3500+ apps** with an average of 7 versions
- Top-down approach: Common anti-patterns with **thresholds** on metrics
- Bottom-up approach: **Machine-learning** algorithms to detect new anti-patterns

**Anti-Patterns Fixing**

- Increase maintainability and understandability
- Reduce energy consumption
- Improve responsiveness
- Decrease package size
- Less bugs

**References**


Geoffrey Hecht <geoffrey.hecht@inria.fr>, Laurence Duchien <laurence.duchien@inria.fr>, Naouel Moha <moha.naouel@uqam.ca>, Romain Rouvoy <romain.rouvoy@inria.fr>