HASLab
HIGH-ASSURANCE
SOFTWARE LABORATORY

HAROS: Quality Assessment of ROS Applications

Integration Workshop at IPA

André Santos

INESC TEC & Universidade do Minho, Portugal

January 31st, 2017

INESCTEC
TECHNOLOGY & SCIENCE
ASSOCIATE LABORATORY
PORTUGAL

Universidade do Minho

Summary

- Many interesting robotics applications (e.g. health, industry) require high levels of safety and flexibility.
- In such scenarios, safety is usually ensured by software.
- This kind of software must be high-quality software.
- Our goal was to get a panorama of the current quality level in many popular ROS robots.
- To achieve this goal, we built a framework to automate the collection of several quality measures.

Software Quality

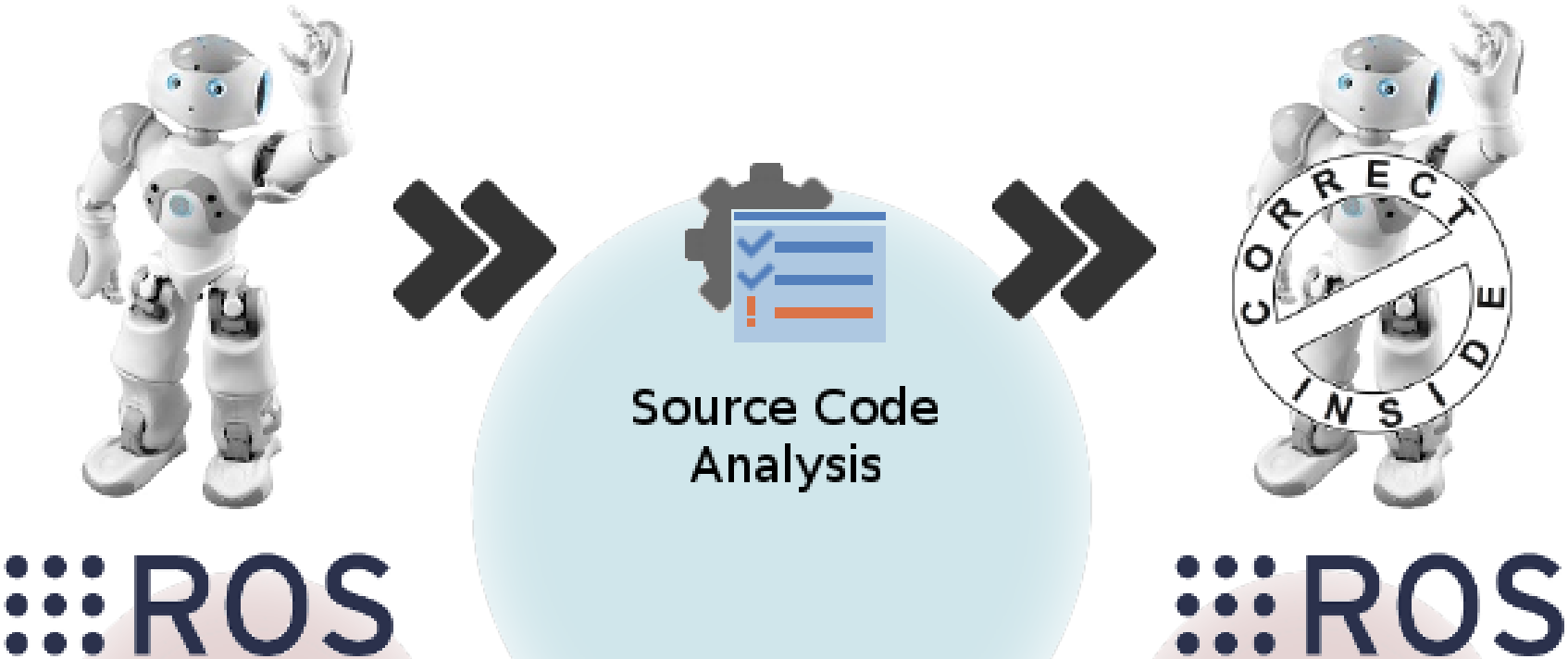
One way to minise safety issues is to produce high-quality software.

To improve safety and quality, adopting coding standards – rules and recommendations about how to write the software – is a common practice (e.g. ROS C++ Style Guide, MISRA C++, HIC++).

Another common technique is to analyse quality metrics – numeric values about how much a property manifests (e.g. lines of code, number of dependencies, function complexity).

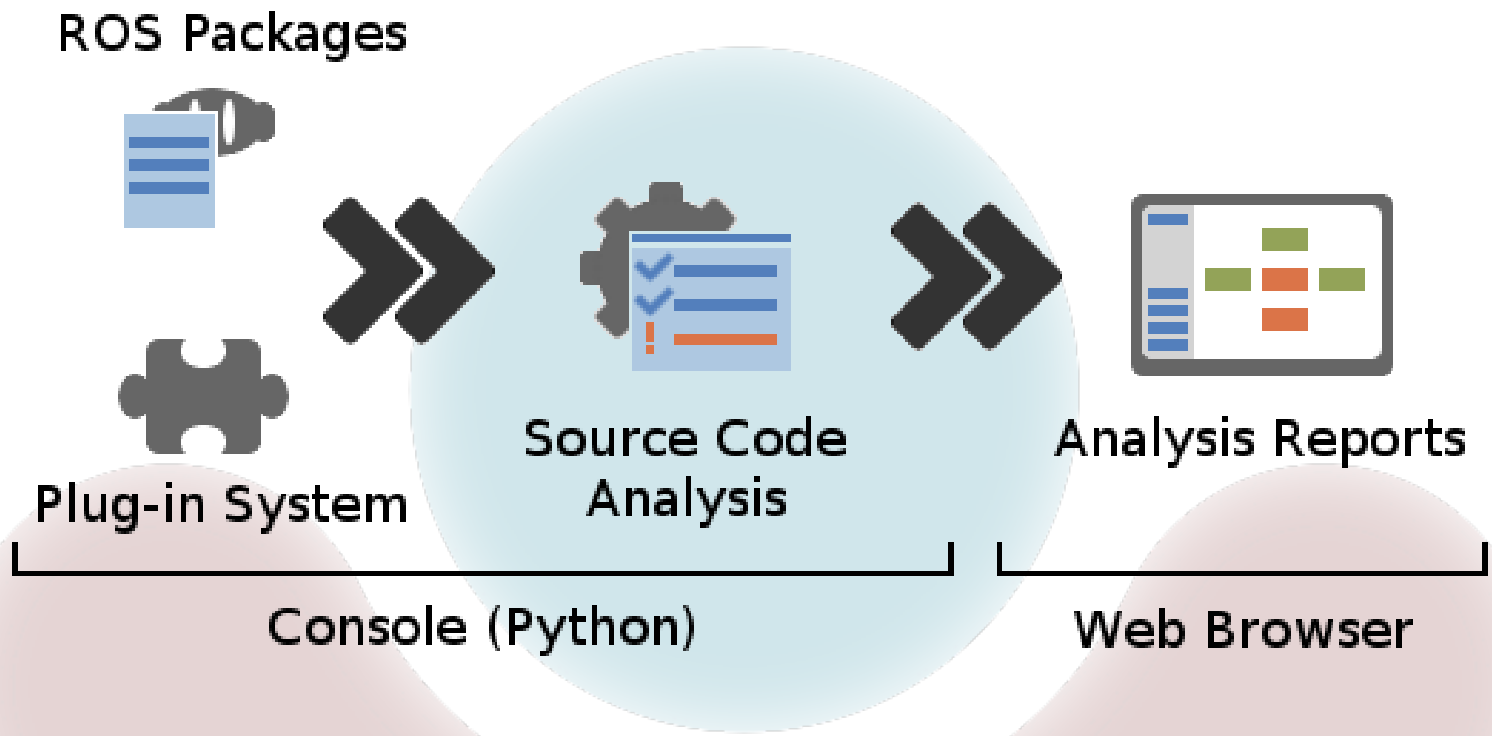
The HAROS Framework: Overview

The HAROS Framework (High-Assurance ROS) aims at providing an analysis platform for ROS systems, making robots more reliable.



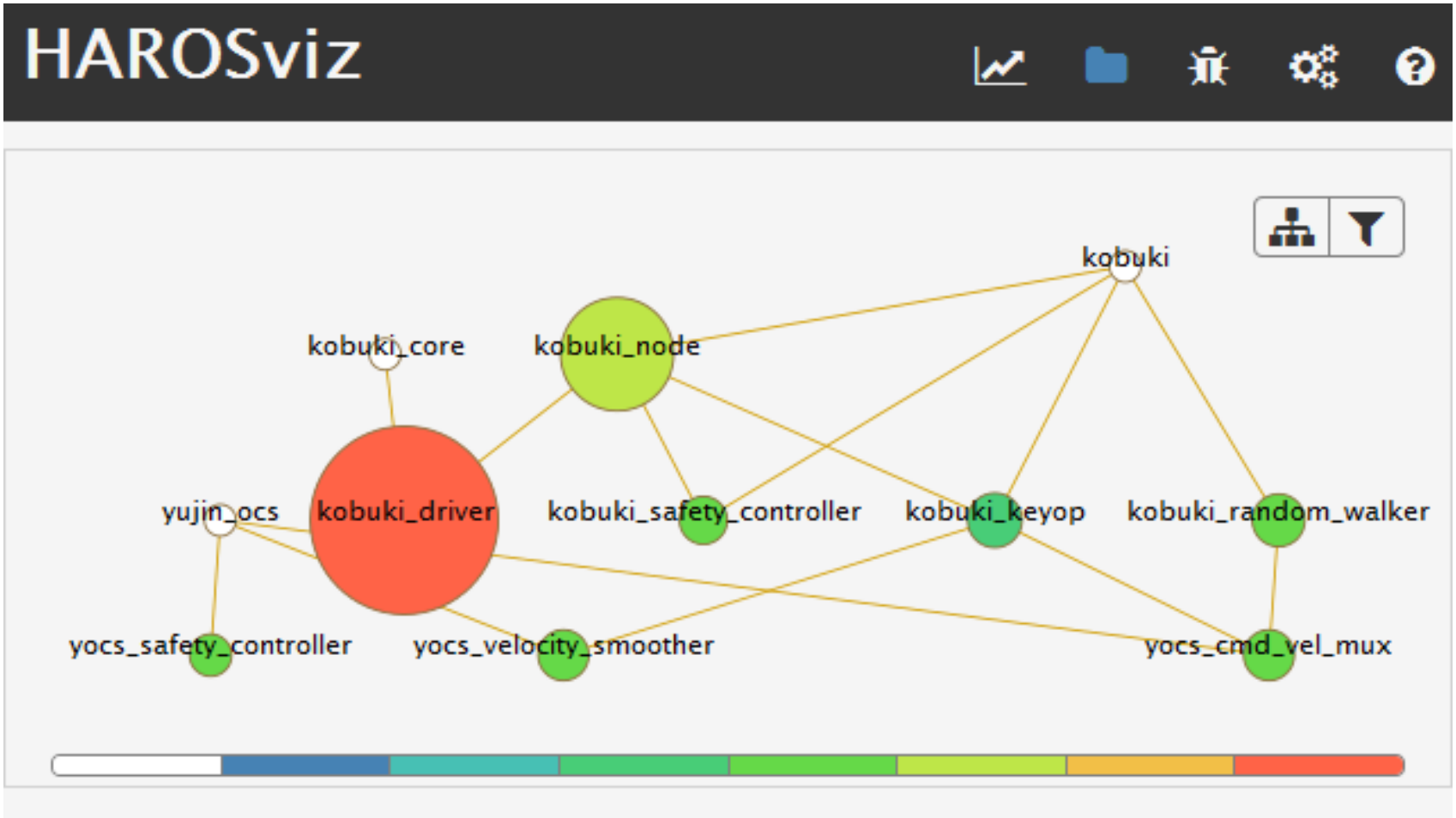
The HAROS Framework: Main Features

- Source code fetching of indexed ROS packages.
- Plug-ins enable integration of third-party analysis tools.
- Interactive graphic reports of the results mirroring the ROS architecture.



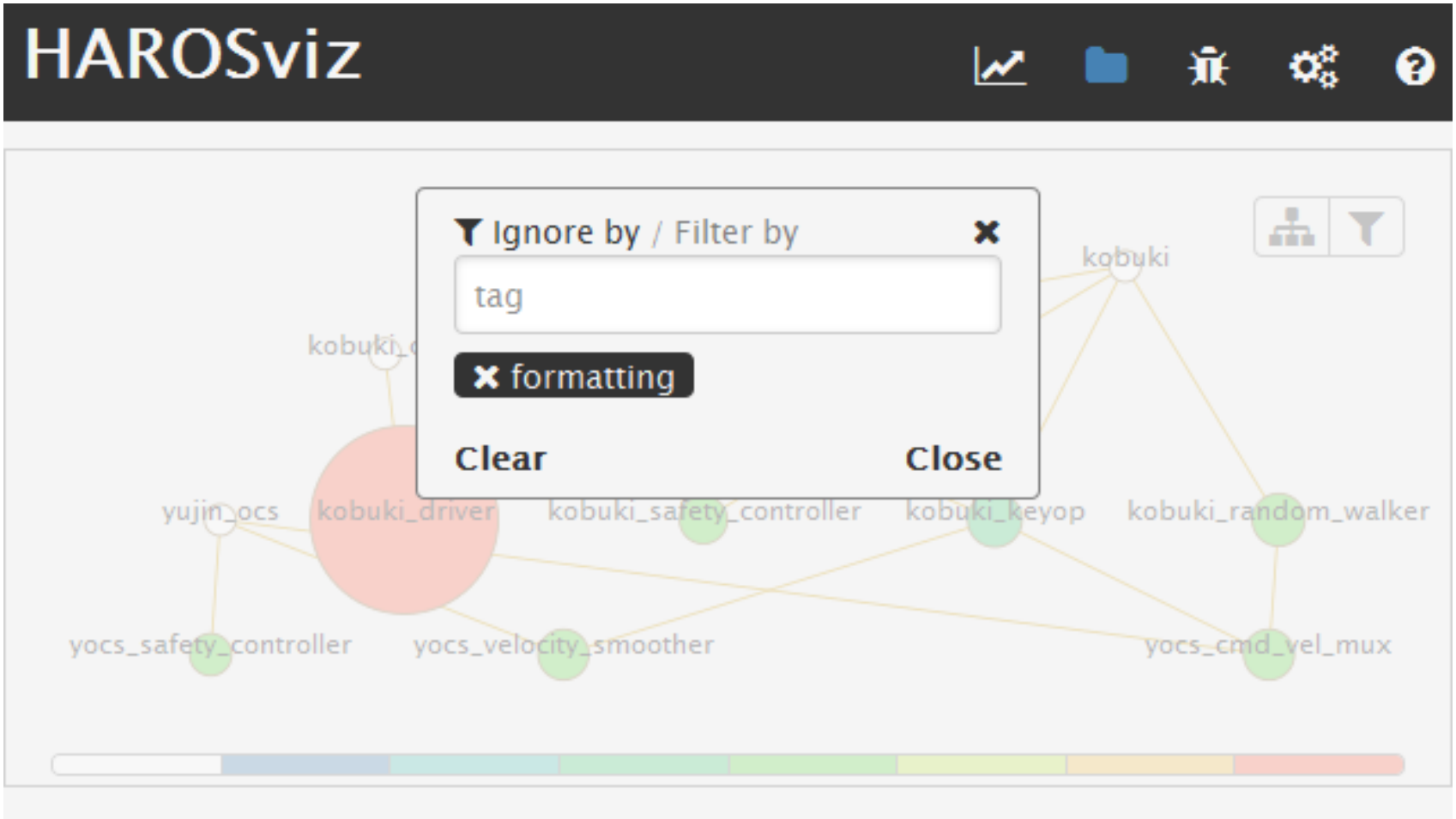
The HAROS Framework: Visualisation

The visualiser builds a diagram of the analysed packages.
Package colours denote the amount of issues.



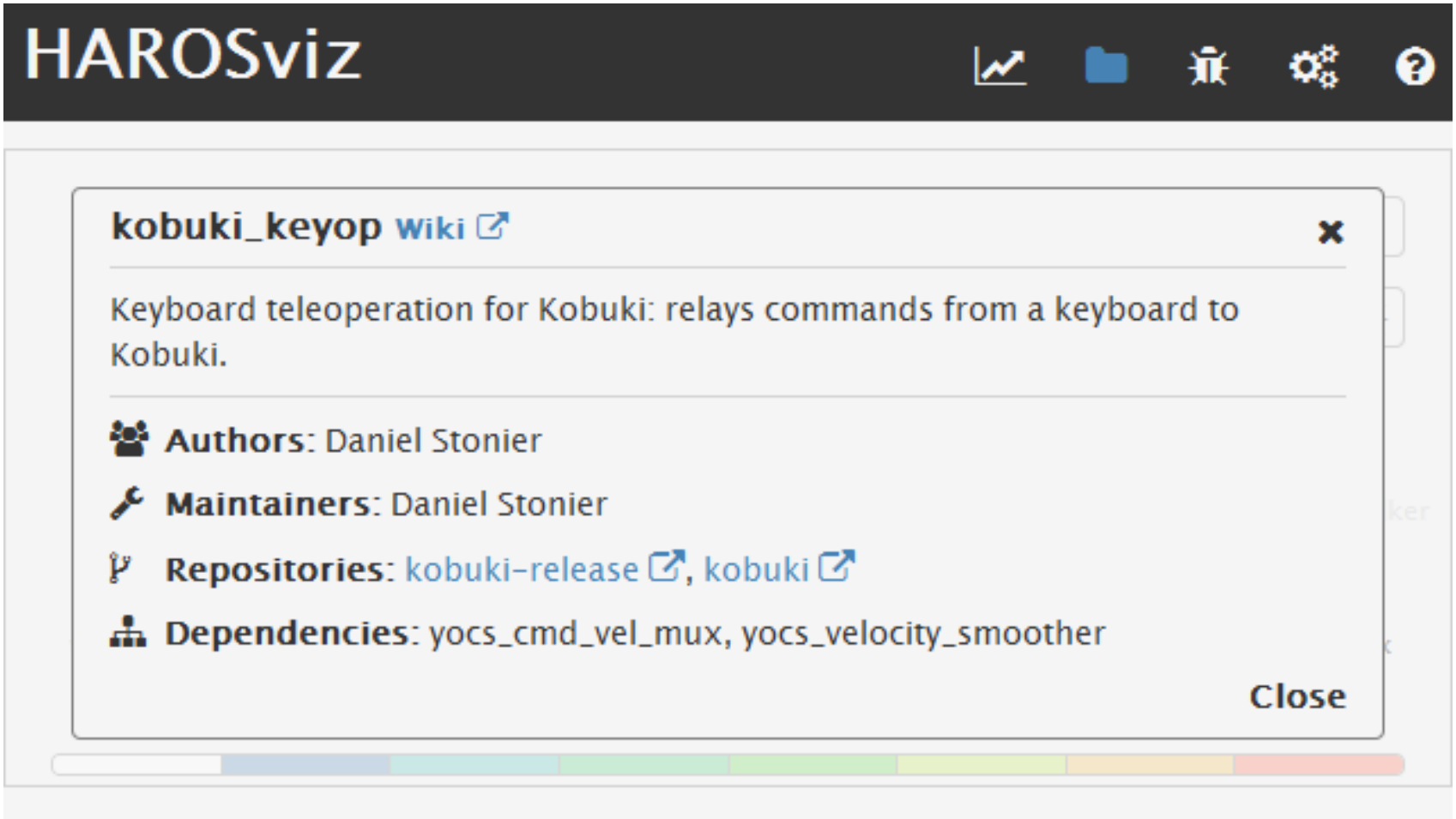
The HAROS Framework: Visualisation

Issues can be filtered or ignored by tags.



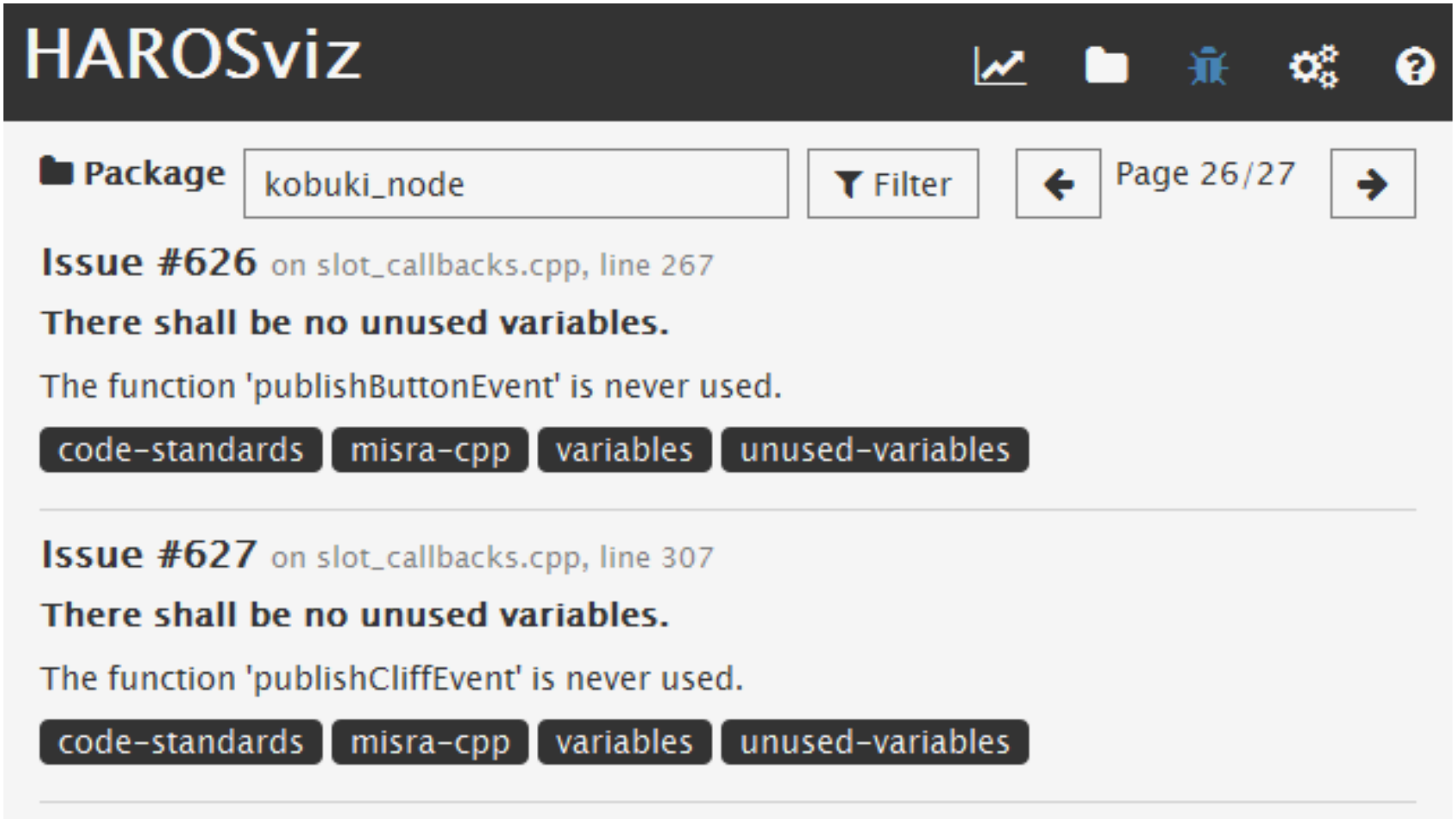
The HAROS Framework: Visualisation

Package details are also available.



The HAROS Framework: Visualisation

Issues can be inspected in detail.



The HAROS Framework: Case Study

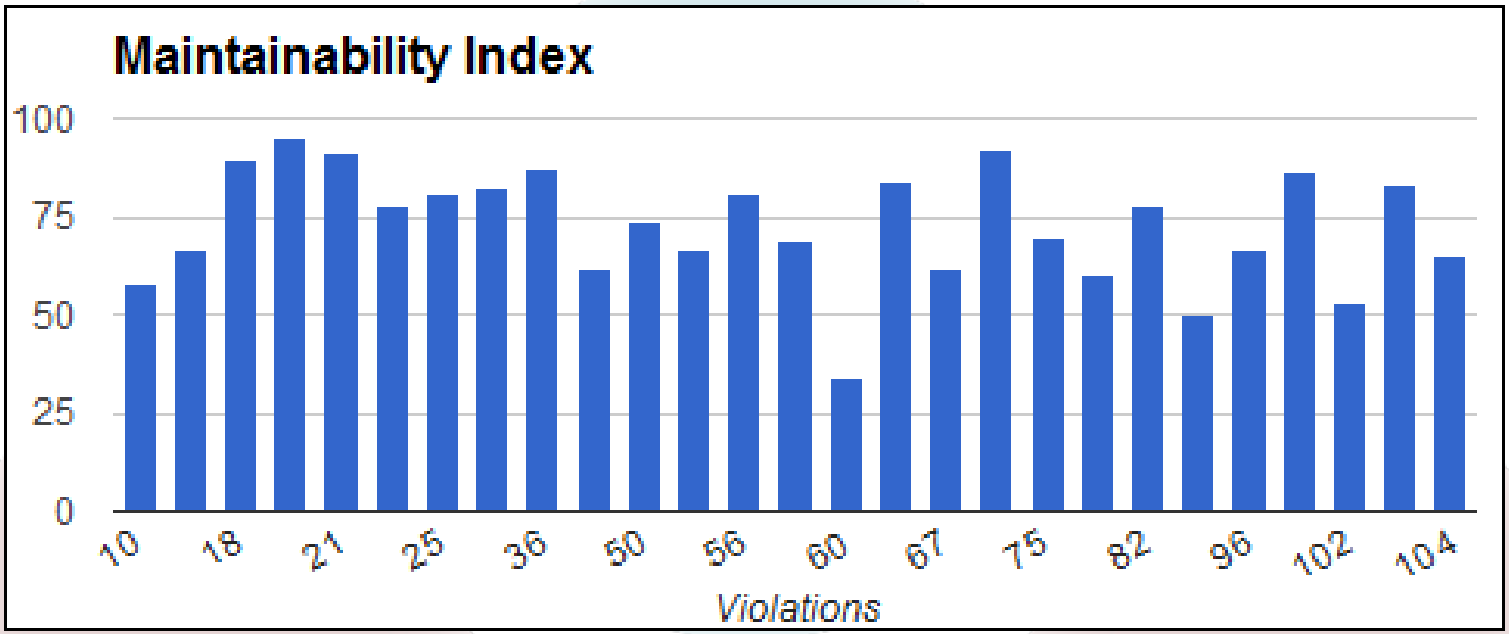
HAROS was applied on a few ROS robots, using CCCC, Radon, Cpplint and Cppcheck as plug-ins.

- Analysis sample: 46 repositories – more than 350 000 lines of C++.
- Assessment of over 100 rules and 15 metrics.
  - Covering ROS and Google's C++ Style Guide, and a small portion of MISRA C++, HIC++, and JSF AV C++.
  - Source metrics: lines of code/comments, comment ratio, maintainability, dependencies, cyclomatic complexity, ...
  - Process metrics (from GitHub): commits, contributors, number of issues.
- Packages categorised as drivers, libraries, or applications.

The HAROS Framework: Case Study

Some observations:

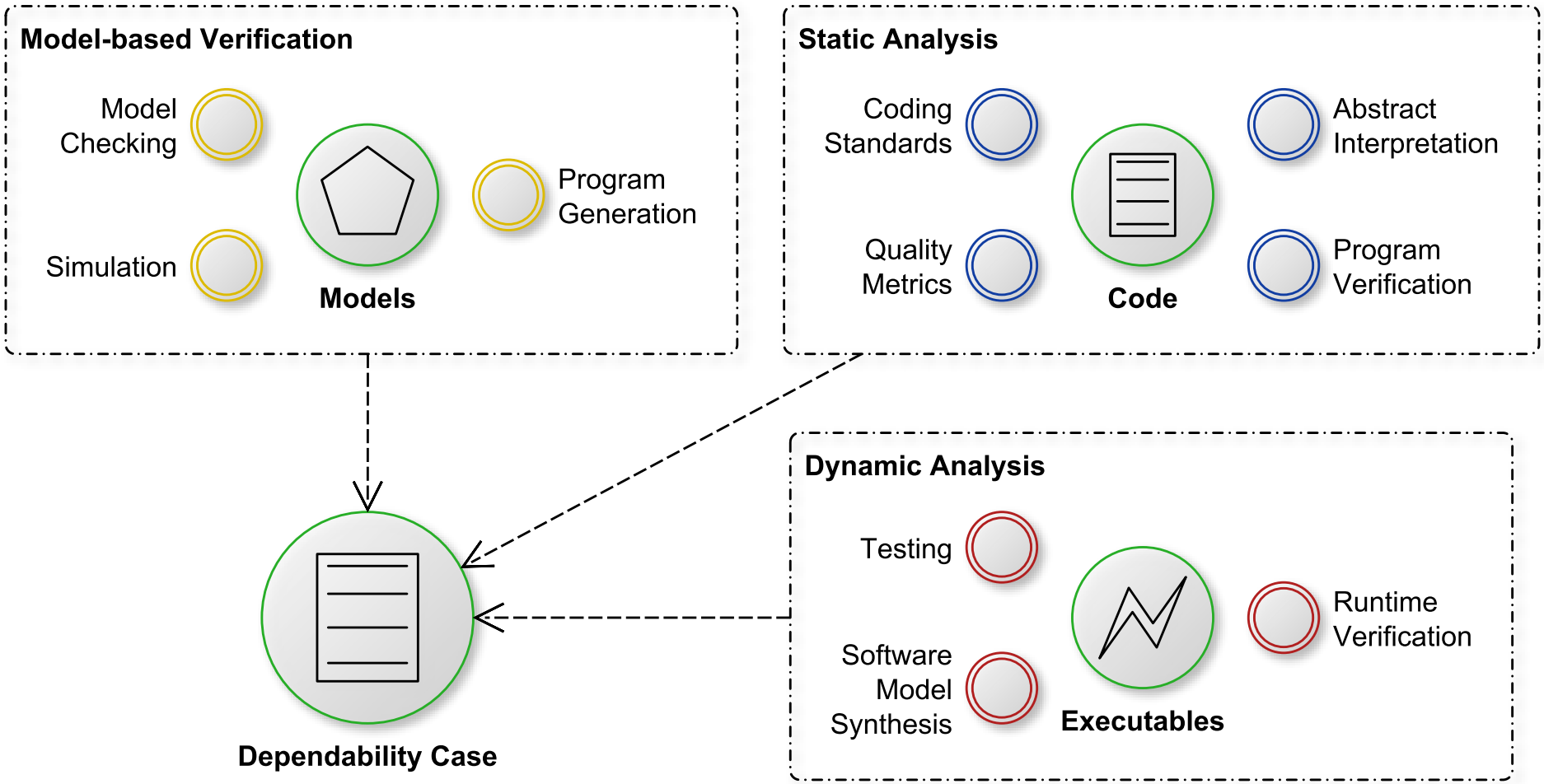
- The projects have thousands of coding rule violations.
- There are few correlations between metrics – the quality is inconsistent.
- Drivers and applications are more active – more developers and commits, but also more issues.



HASLab
HIGH-ASSURANCE SOFTWARE LABORATORY

What's Next: Software Safety, Better Quality

There are various techniques to assess software safety and quality.



Research Questions

During our reserach, we hope to answer the following questions.

1. How to define a domain specific language for specifying safety properties over ROS code?

2. Is it possible to identify a catalogue of generic safety properties that apply to most ROS systems?

3. How to define model extraction techniques that reverse engineer ROS code into abstract models?

4. How to define ROS coding standards and best practices that make such model extraction feasible?

HAROS: The NYI List

- Ranking of issues (e.g. by severity).
- Providing ROS-specific metrics.
- Dedicated view for raw metrics.
- Continuous tracking of package quality.
- Integration with catkin.

Give HAROS a try at
git-afsantos.github.io/haros or github.com/git-afsantos/haros

IROS 2016 paper at
haslab.uminho.pt/afsantos/publications

Reach me at
haslab.uminho.pt/afsantos

Thank you!

HASLab
HIGH-ASSURANCE
SOFTWARE LABORATORY