ROS in Actual Practice
Mining Usage Patterns of Its Primitives

ROS-Industrial Conference 2017

André Santos
Alcino Cunha    Nuno Macedo

HASLab, INESC TEC & University of Minho, Portugal

December 13th, 2017
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.

› Automated software analysis can detect a considerable range of bugs and reduce the time spent on maintenance.

› Our goal is to apply and design verification techniques on the robotics domain – focusing on ROS applications.

› Our prototype framework provided us an overview of the current coding practices, and usage statistics of several ROS features.
Software Analysis

Software analysis can be done at different abstraction levels.

Each technique has its benefits and limitations.
Software Analysis

Our work explores mostly static analysis.

One of our goals is to reverse engineer code into formal models.
Challenges of ROS Static Analysis

1. Complex programming languages, e.g. C++.

2. No well-defined concept of application.
   Launch files are the closest thing we get.

3. Internal node behaviour is not enough.
   Inter-node communication also needs to be analysed.

4. Use of conditionals and external variables in launch files.

5. Multiple variants (overloads) for the same ROS primitives.

6. No coding standards widely adopted by the community.
The HAROS Framework

HAROS (High-Assurance ROS) aims at providing an analysis platform for ROS systems.

It features online package fetching, a plug-in system and interactive analysis reports.
The HAROS Framework

The visualiser provides a number of general and ROS-specific statistics.

<table>
<thead>
<tr>
<th>Source Code</th>
<th>ROS Components</th>
<th>Quality Progress</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packages</td>
<td>Launch files</td>
<td>Avg. Function Length</td>
</tr>
<tr>
<td>Files</td>
<td>Configurations</td>
<td></td>
</tr>
<tr>
<td>Scripts</td>
<td>Node count</td>
<td></td>
</tr>
<tr>
<td>C++</td>
<td>Nodelets</td>
<td></td>
</tr>
<tr>
<td>96% - 0%</td>
<td>Parameter files</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Issues</td>
<td>ROS Communications</td>
<td></td>
</tr>
<tr>
<td>Total issues</td>
<td>Topics</td>
<td></td>
</tr>
<tr>
<td>Coding standards</td>
<td>Remappings</td>
<td></td>
</tr>
<tr>
<td>Code metrics</td>
<td>Custom messages</td>
<td></td>
</tr>
<tr>
<td>Other issues</td>
<td>Custom services</td>
<td></td>
</tr>
<tr>
<td>Issues per line</td>
<td>Custom actions</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>0.06</td>
<td>n/a</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The HAROS Framework

The visualiser also builds a diagram of package dependencies.

Package colours denote the amount of analysis issues.
Mining Usage Statistics: ROS Features

We used feature diagrams to enumerate and organise the various features and components of a ROS application.
Two independent modules analyse C++ and launch files. A manager component connects the two and tries to recreate the computation graph.
Mining Usage Statistics: Analysed Corpus

We performed this analysis on 419 packages, consisting of 365 applications and 929K C++ LoC.
Mining Usage Statistics: C++ Files

Client / Server
- 82% Publisher / Subscriber
- 18%

General Communications
- 32% Custom Messages
- 88% Method Callbacks
- 74% Literal Topics
- 15% Global Names

Parameter Server
- 89% of Packages
- 95% Get Param
- 83% Get With Default

Static Analysis Coverage
- Packages
  - 69%
  - 80%
  - 71%
- Supporting
  - Just Literals
  - Literals and Variables
  - Literals and Control Flow
- Applications
  - 24%
  - 30%
  - 31%

André Santos
Mining Usage Statistics: Launch Files

1418 Nodes
- 19% Unique
- 8% Nodelets
- 3% Required
- 39% Respawn

3735 Parameters
- 26% from YAML
- 6% from Commands

46 Machines
- assigned to localhost
- used to run scripts

365 Applications
- 4 Nodes
- 11 Parameters
- 3 Includes
- 3 Remappings
- 2 Env. Variables
- 3 Conditionals
Final Remarks

› Understanding of how ROS software is developed and used in practice.

› This could lead to updates in guidelines and tutorials.

› Helps tool developers identify where to invest their effort.

› A straight-forward analysis tool is enough for many use cases.

› Increasing coverage requires sophisticated static analysis.
Ongoing Work: HAROS v3.0

Our current focus is on refining the metamodel and updating the extractors to match it.
Ongoing Work: HAROS v3.0

› Make the extractors available as a core feature of HAROS.

› Provide a graphical visualisation for the extracted models.

› Implement a query language that operates on the extracted model.
Future Work

- Expand our analysis database, both in sample size and features.
- Come up with coding rules that ease the analysis and help avoid common bugs.
- Explore specification languages to complement the extracted models.
Analysis tools at
github.com/git-afsantos/haros    github.com/git-afsantos/bonsai

Online data set at
github.com/git-afsantos/ros_data

IROS papers at
haslab.uminho.pt/afsantos/publications

Thank you!