
# High-assurance Robotics Software

André Santos  Nuno Macedo  Cláudio Lourenço

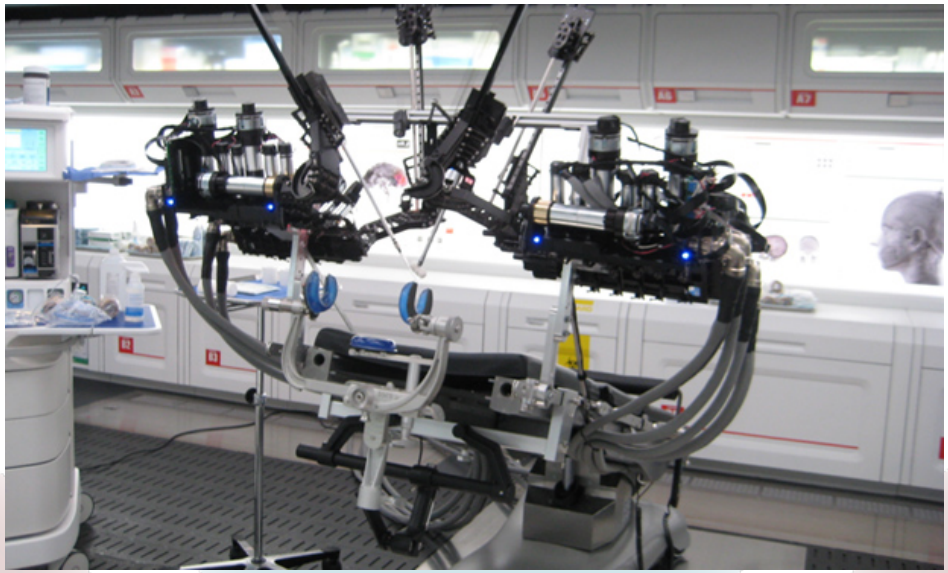
HASLab / INESC TEC & Universidade do Minho, Portugal

October 14th, 2015


# Why robotics?

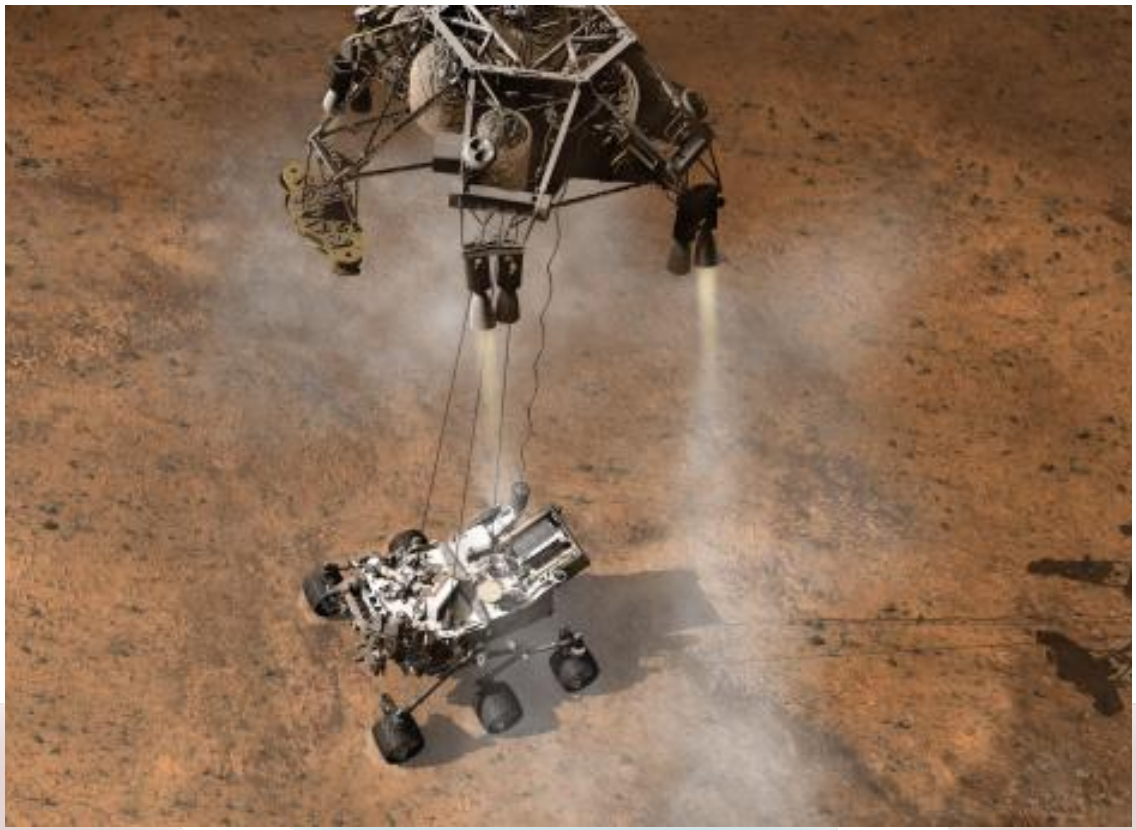
Robots began as electronic analog systems, used for simple tasks. Nowadays, robots are digital, complex, autonomous, often expensive, and they interact with people, or replace people in various activities.

Robots are now used even in safety-critical applications, such as health and industrial devices. This makes robot quality assurance a priority.



# Why robotics?

Curiosity, NASA's Mars Rover, is a successful case where software formal verification has been applied in robotics.



# Why robotics?

## Robot vacuum cleaner 'attacks' South Korea housewife's hair

Woman sleeping on floor of her apartment awoken in pain by robotic vacuu cleaner attempting to suck up her hair



The vacuum only stopped running after initially ingesting the woman's hair Photo: khan news

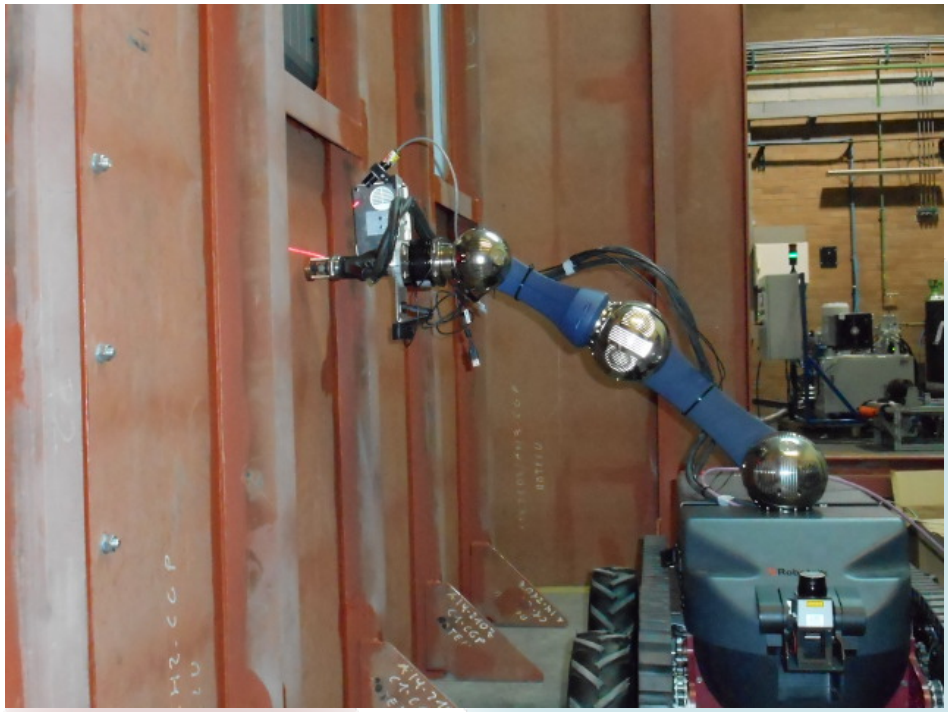
In *The Telegraph*

# Why robotics?

CARLoS, a shipbuilding robot in which INESC TEC was involved.

# Why robotics?

This robot works in a safety-critical environment: ships must not sink due to poor manufacture, and people must not be harmed.



# Why ROS?

The Robot Operating System is a set of open source software libraries and tools to build robot applications. It covers all abstraction layers, from hardware drivers, to complex algorithms.

ROS is developed in common programming languages, such as C++ and Python. Its community stands now on tens of thousands of users, and it is used in industry, research and education.


# Why ROS?

Some examples of robots that support ROS:

# Research Overview

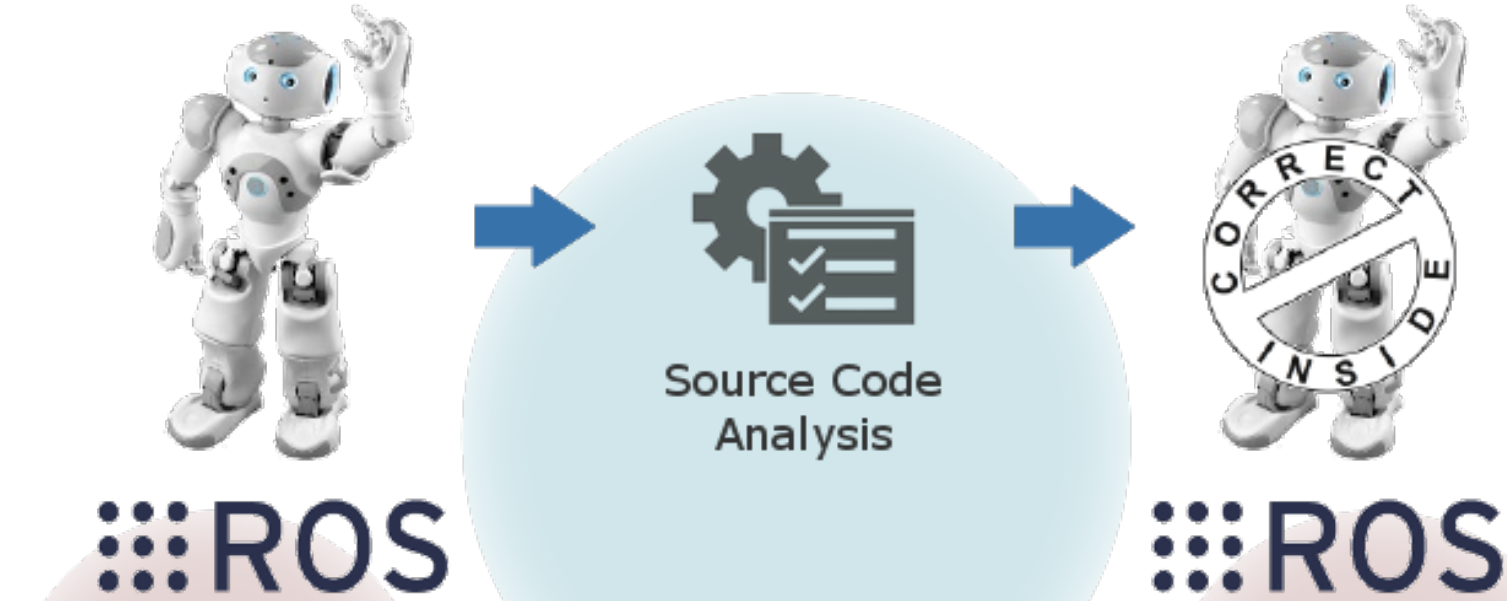
The main objective is to understand how software quality measurement techniques can be applied to ROS software, in order to improve its overall quality. Our work includes:

- studying and comparing analysis techniques and analysis tools, with emphasis on high-reliability systems;
- producing extensible ROS-specific tools to analyse ROS software;
- using the new tools to assess code quality of existing ROS systems.

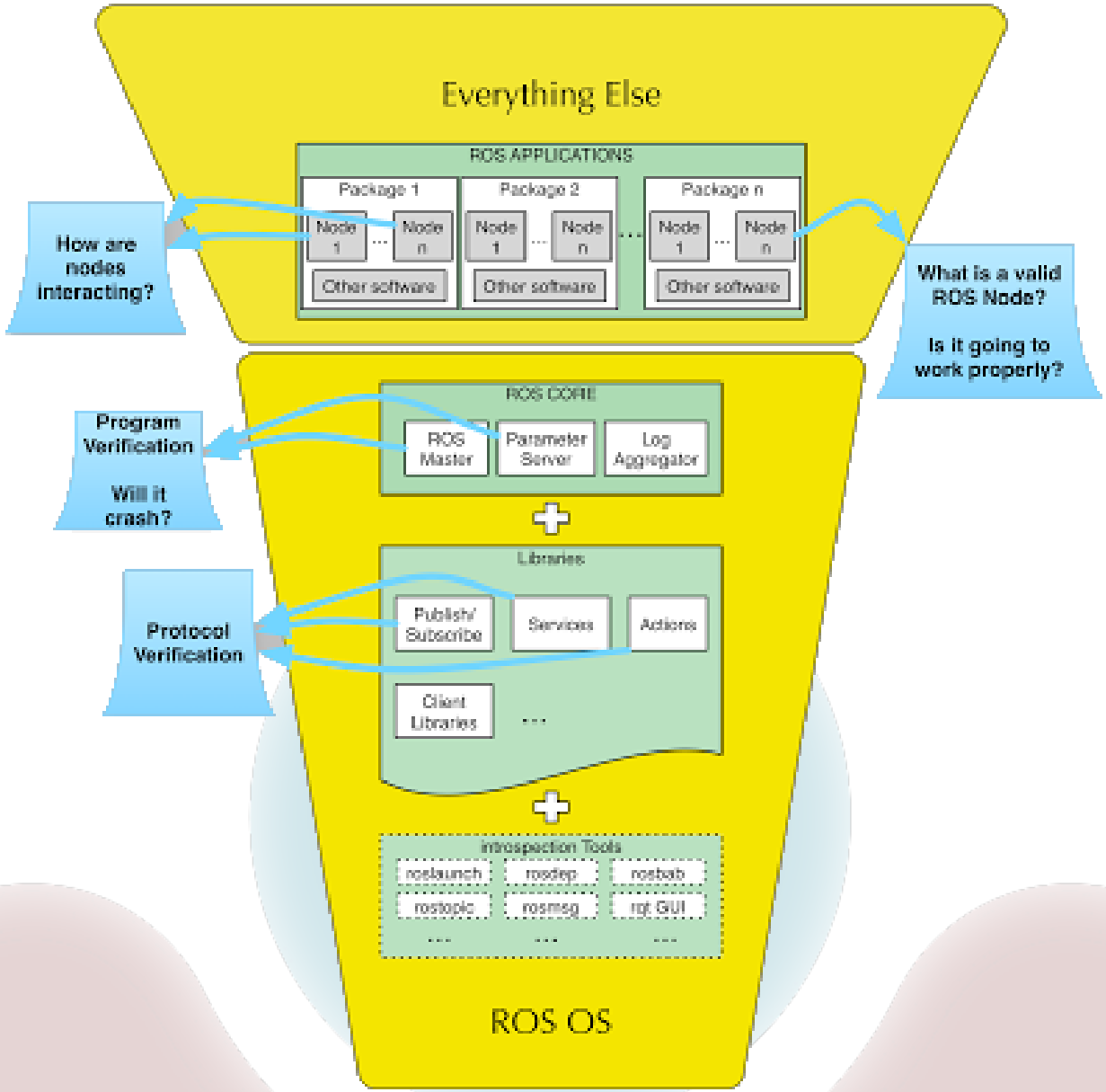
# Research Overview

In other words, we use (and develop) a set of tools capable of performing static analysis on ROS software and producing analysis reports.

Verifying coding rules, gathering code metrics, among other techniques, lead to reliable software, and thus to reliable robots.



# Research Overview






HASLab
HIGH-ASSURANCE SOFTWARE LABORATORY

# Code Metrics

Metrics are a common analysis technique. Some are simple enough for the regular developer to understand – *e.g.* Source Lines of Code, Comment Ratio – while others are quite more convoluted – *e.g.* McCabe's Cyclomatic Complexity, Halstead's Programming Effort.

While some metrics are more useful and accurate than others, they are often used to:

- Assess the overall quality of a project – How much effort and cost is required to maintain it?
- Control the progress of a project – Are patches actually improving the code?
- Estimate the number of bugs left in the program and predict component failures.

# Coding Standards

Coding standards define a set of rules to preemptively avoid unreadable, inconsistent and unsafe code. They often categorise their rules by topic and compliance level.

Widely adopted standards, such as MISRA C++, HIC++ or JSF AV C++ are strict, well documented and focused on reliable and safe software. Others, such as Google C++ and ROS C++ act more as style guides – they focus more on formatting or naming issues.

There are a number of very capable commercial tools to verify compliance with the stricter standards. Free tools do little *out of the box*, in terms of verification. They provide extensibility instead, and let their users implement additional checks.

# Ongoing Research

## Program Verification

- Allows verification of functional properties.
- Ensures code free of bugs.

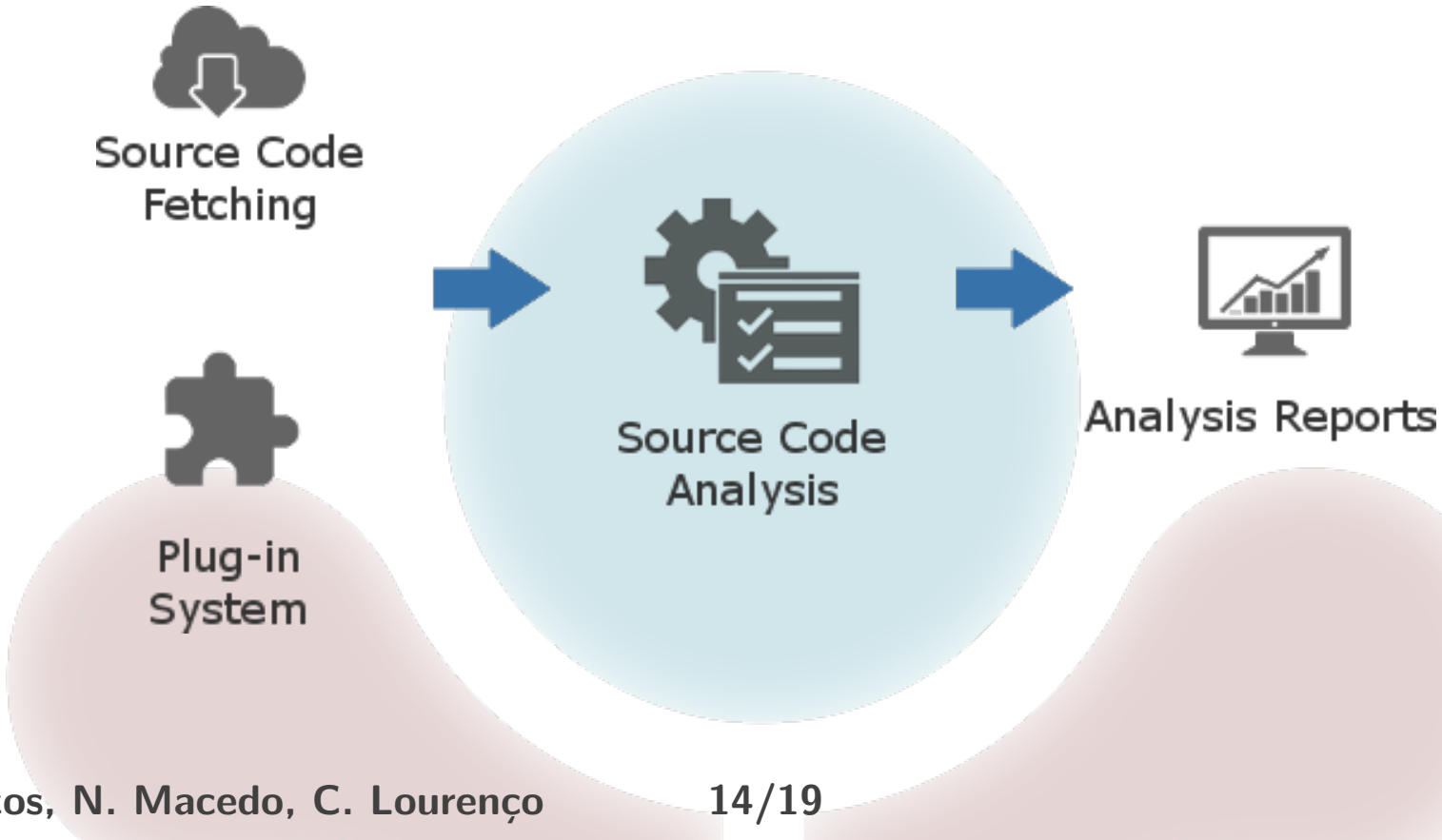
## Model Verification

- Provides abstractions of architectures and algorithms.
- Allows discovery of reasoning and design fallacies.

# A ROS Static Analysis Tool

Static analysis needs automated tools. The existing free C++ tools are few, and the commercial tools do not target the specifics of a ROS system.

This project proposes a new ROS specific static analysis tool, one that is generic, extensible and capable of reusing existing tools.






HASLab
HIGH-ASSURANCE
SOFTWARE LABORATORY

# Case Study

We applied the new tool to existing ROS applications. We selected 11 relevant robots made with ROS to analyse in terms of software quality, with emphasis on code metrics.

In order to perform this analysis, we reused existing analysis tools as plug-ins for our new tool. Popular tools such as CCCC, Radon, Cpplint and Cppcheck were integrated into our tool.

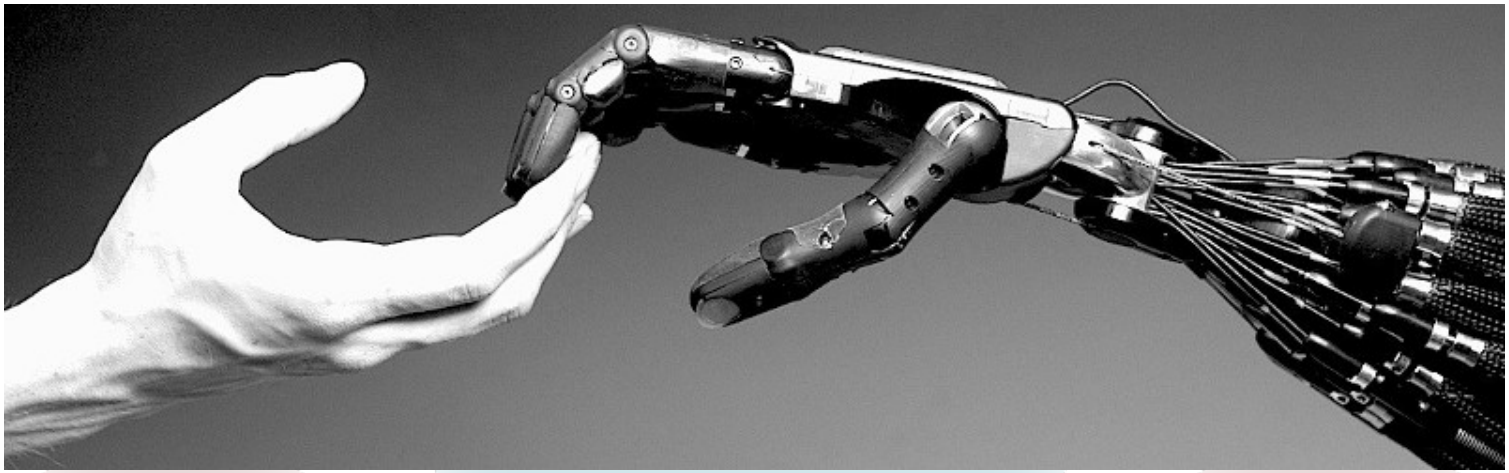
# Case Study

Here follow some relevant analysis aspects and results.

- The analysis sample consists of 46 GitHub repositories – more than 350 000 lines of C++ code;
- The plug-ins verify compliance with over 100 rules;
- We are currently considering more than 15 code metrics – for instance, McCabe's Cyclomatic Complexity and Halstead's Volume;
- In general, the projects have thousands of coding rule violations;
- There are few correlations between the extracted metrics – the quality is inconsistent.

# Final Remarks

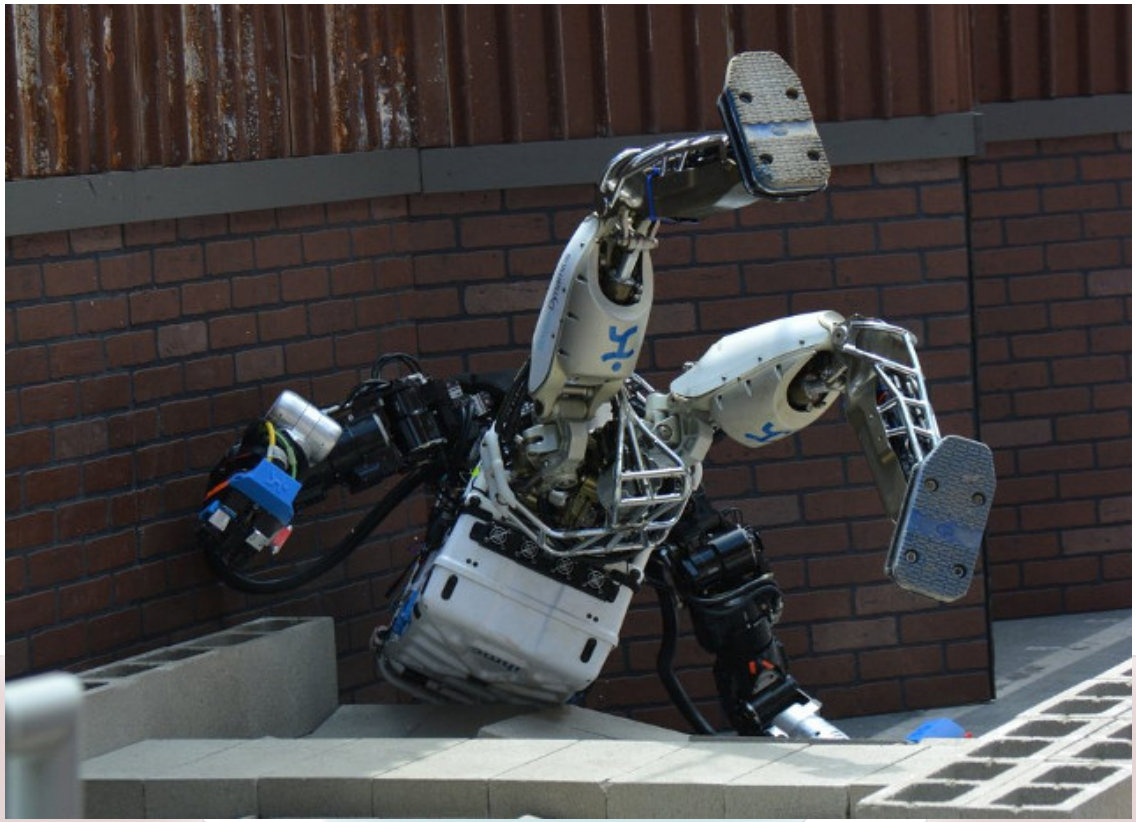
- Robotics is a significant field of research, now and in the foreseeable future.
- ROS and other open source robotics frameworks pushed the limits of robotics research.



# Final Remarks

However...

Quality assurance of robotics software is still lacking, in contrast to other software applications.



# Questions?