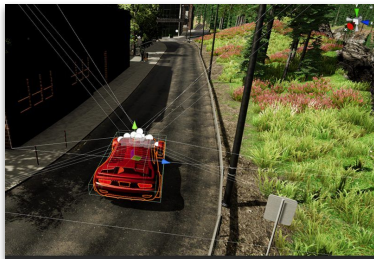


188* ROS bugs later: Where do we go from here?

Christopher Timperley

ROSCon 2019, Macau

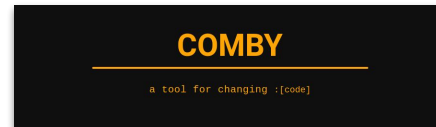
ROS



squaresLab
Software Quality in
Real Evolving Systems

isr institute for
SOFTWARE
RESEARCH

**Carnegie
Mellon
University**



- program repair
- fuzzing
- testing
- program transformation

An unlikely beginning...

An unlikely beginning...



39th International Conference
on Software Engineering

May 20-28, 2017 - Buenos Aires, Argentina



39th International Conference
on Software Engineering

May 20-28, 2017 - Buenos Aires, Argentina



39th International Conference on Software Engineering

May 20-28, 2017 - Buenos Aires, Argentina

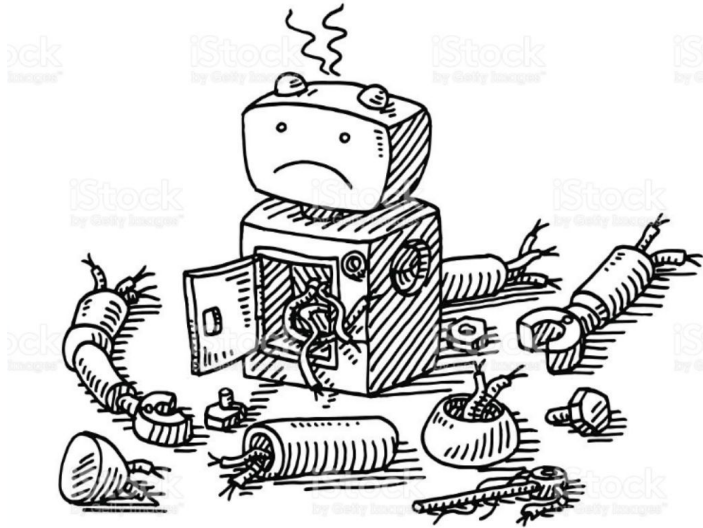


39th International Conference on Software Engineering

May 20-28, 2017 - Buenos Aires, Argentina



What kinds of bugs do ROS developers encounter?



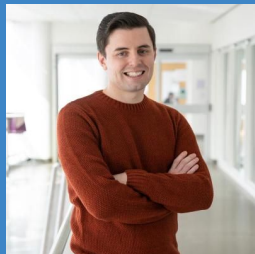
Understanding ROS bugs



Tools, Tips, and Techniques

Meet the ROBUST team!

Program
Repair



Software
Quality

Static
Analysis

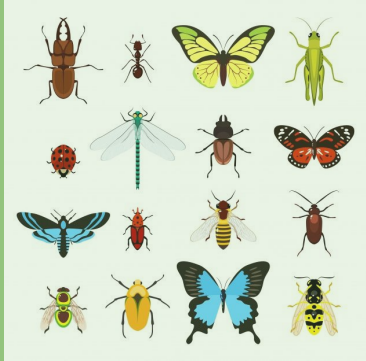


Robotics

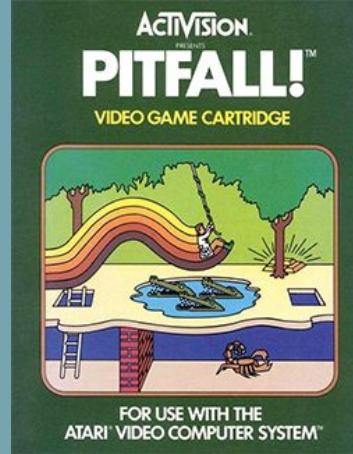




ROBUST: ROS Bug Study



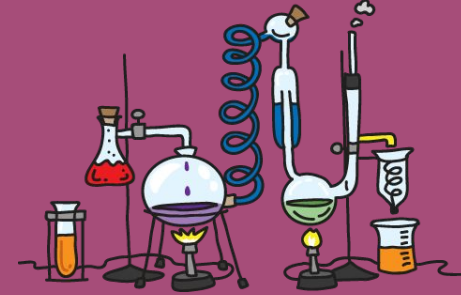
How we
collected
our bugs



Common
developer
pitfalls



Improve
your ROS
code today



Building
better ROS
code for
tomorrow

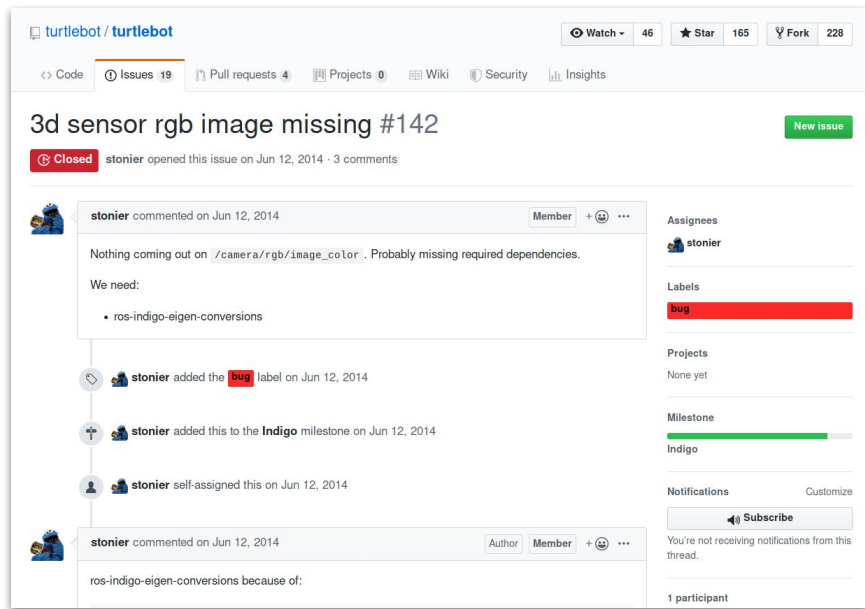
We studied ~~188~~ 266 bugs across 8 ROS packages

Subject	# Bugs
MAVROS	57
Kobuki	40
Universal Robot	25
Motoman	22
TurtleBot	12
Care-O-Bot	11
geometry2	36
<i>Confidential</i>	10



We analysed 1790 GitHub issues and pull requests

Subject	# Bugs	# Issues
MAVROS	57	325
Kobuki	40	623
Universal Robot	25	158
Motoman	22	78
TurtleBot	12	170
Care-O-Bot	11	182
geometry2	42	254
<i>Confidential</i>	10	N/A



We forensically described each bug

turtlebot / turtlebot

Watch 46 Star 165 Fork 228

Code Issues 19 Pull requests 4 Projects 0 Wiki Security Insights

3d sensor rgb image missing #142

Closed stonier opened this issue on Jun 12, 2014 · 3 comments

stonier commented on Jun 12, 2014

Nothing coming out on `/camera/rgb/image_color`. Probably missing required dependencies.

We need:

- ros-indigo-eigen-conversions

stonier added the **bug** label on Jun 12, 2014

stonier added this to the **Indigo** milestone on Jun 12, 2014

stonier self-assigned this on Jun 12, 2014

stonier commented on Jun 12, 2014

ros-indigo-eigen-conversions because of:

Assignees: stonier

Labels: **bug**

Projects: None yet

Milestone: Indigo

Notifications: **Subscribe**

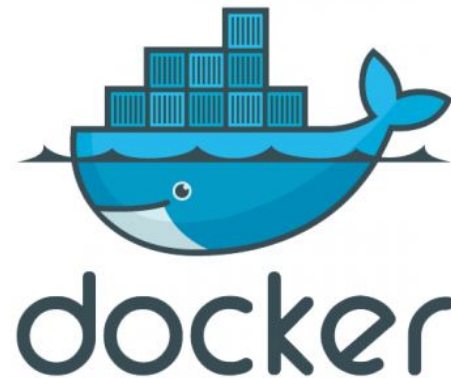
You're not receiving notifications from this thread.

1 participant

```
id: f01d952
title: No Image Coming From Camera
description: >
  A missing runtime dependency crashed the image processing node,
  causing no images to be received from the camera.
classification: Missing Dependency (no CWE)
keywords: ['camera', 'eigen', 'dependencies', 'runtime']
system: turtlebot
severity: error
bug:
  phase: runtime-operation
  specificity: ROS-specific
  architectural-location: application-specific code
  application: null
  task: vision
  subsystem: driver
  package: ros-perception/image_pipeline/depth_image_proc
  languages:
    - C++
  detected-by: runtime crash
  reported-by: contributor
  issue: https://github.com/turtlebot/turtlebot/issues/142
  time-reported: 2014-06-12 (10:04)
  reproducibility: always
trace: |
  [ INFO] [1402564815.530736554]: /camera/rgb/camera_info -> /camera/rgb/camera_info
  [ERROR] [1402564815.727176562]: Failed to load nodelet [/camera/depth_metric_rect] of type
  [depth_image_proc/convert_metric]: Failed to load library
  /opt/ros/indigo/lib/libdepth_image_proc.so.
  Make sure that you are calling the PLUGINLIB_EXPORT_CLASS macro in the library code, and that
  ...
```

We built a Docker image for each bug

```
id: f01d952
title: No Image Coming From Camera
description: >
  A missing runtime dependency crashed the image processing node,
  causing no images to be received from the camera.
classification: Missing Dependency (no CWE)
keywords: ['camera', 'eigen', 'dependencies', 'runtime']
system: turtlebot
severity: error
bug:
  phase: runtime-operation
  specificity: ROS-specific
  architectural-location: application-specific code
  application: null
  task: vision
  subsystem: driver
  package: ros-perception/image_pipeline/depth_image_proc
  languages:
    - C++
  detected-by: runtime crash
  reported-by: contributor
  issue: https://github.com/turtlebot/turtlebot/issues/142
  time-reported: 2014-06-12 (10:04)
  reproducibility: always
  trace: |
    [ INFO] [1402564815.530736554]: /camera/rgb/camera_info -> /camera/rgb/camera_info
    [ERROR] [1402564815.727176562]: Failed to load nodelet [/camera/depth_metric_rect] of type
    [depth_image_proc/convert_metric]: Failed to load library
    /opt/ros/indigo/lib/libdepth_image_proc.so.
    Make sure that you are calling the PLUGINLIB_EXPORT_CLASS macro in the library code, and that
    ...
```



We used a *time machine* to build historically-accurate Docker images

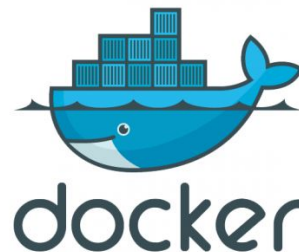


rosinstall_generator_time_machine



```
- tar:  
  local-name: actionlib  
  uri: https://github.com/ros-gbp/actionlib-...  
  version: actionlib-release-release-indigo-actionlib-1.11.3-0  
- tar:  
  local-name: angles  
  uri: https://github.com/ros-gbp/geometry_angles...  
  version: geometry_angles_utils-release-release-indigo-angles-1.9.9-0  
- tar:  
  local-name: catkin  
  uri: https://github.com/ros-gbp/catkin-release/archive/...  
  version: catkin-release-release-indigo-catkin-0.6.14-0  
...
```

deps.rosinstall



Try it out for yourself: <https://bit.ly/2NnkmSr>



What kinds of bugs do ROS developers encounter?

More than a few...

Incorrect type casts and conversions
Missing or incorrect frame conversions
Missing build-time dependencies
Missing run-time dependencies
Using the wrong robot model
Unintentional use of defaults
Incorrect use of namespacing
Incorrect mathematical operations
Lack of synchronisation leads to race condition
Bad name remappings
Incorrect constant values
Hardcoded values lead to silent errors
Encoding errors
Circular dependencies
Buffer overflow
API changes
Ignoring stop instruction
Inconsistent naming
Missing input validation
Missing variable initialisation
Incorrect data parsing
Python run-time errors
Use of obsolete functions
Incorrect type casts and conversions
Missing or incorrect frame conversions

Circular dependencies
Buffer overflow
API changes
Ignoring stop instruction
Inconsistent naming
Missing input validation
Missing variable initialisation
Incorrect data parsing
Python run-time errors
Function call with incorrect arguments
Uncaught exceptions
Infinite loops
Off-by-one errors
Use of global rather than local namespace
Use of obsolete functions
Incorrect type casts and conversions
Missing or incorrect frame conversions
Missing build-time dependencies
Missing run-time dependencies
Using the wrong robot model
Unintentional use of defaults
Incorrect use of namespacing
Incorrect mathematical operations
Lack of synchronisation leads to race condition
Bad name remappings
Incorrect constant values
Hardcoded values lead to silent errors
Encoding errors

Off-by-one errors
Use of global rather than local namespace
Use of obsolete functions
Incorrect type casts and conversions
Missing or incorrect frame conversions
Missing build-time dependencies
Missing run-time dependencies
Using the wrong robot model
Unintentional use of defaults
Incorrect use of namespacing
Incorrect mathematical operations
Lack of synchronisation leads to race condition
Bad name remappings
Incorrect constant values
Hardcoded values lead to silent errors
Encoding errors
Incorrect type casts and conversions
Missing or incorrect frame conversions
Missing build-time dependencies
Missing run-time dependencies
Using the wrong robot model
Unintentional use of defaults
Incorrect use of namespacing
Incorrect mathematical operations
Off-by-one errors
Use of global rather than local namespace
Use of obsolete functions
Incorrect type casts and conversions

1. Missing system build dependencies!

```
<?xml version="1.0"?>
<package format="2">
  <name>libmavconn</name>
  <version>0.9.0</version>
  <description>
    MAVLink communication library.
    This library provide unified connection
    handling classes and URL to connection
    object mapper.
    ...
  </description>
  ...

  <buildtool_depend>catkin</buildtool_depend>
  <depend>boost</depend>
  <depend>mavlink</depend>
+ <depend>libconsole-bridge-dev</depend>
  <test_depend>gtest</test_depend>

  ...
</package>
```

id: [dc1327f](#)
title: Missing mavlink dependency
system: mavros
description: >
Versions 0.9 and 0.8.3 of MAVROS fail to build due to a missing "mavlink" dependency.

```
-- ~~~~~
-- ~ traversing 1 packages in topological order:
-- ~ - libmavconn
-- ~~~~~
-- +++ processing catkin package: 'libmavconn'
-- ==> add_subdirectory(repo-under-test/libmavconn)
CMake Error at repo-under-test/libmavconn/CMakeLists.txt:10 (find_package):
  By not providing "Findconsole_bridge.cmake" in CMAKE_MODULE_PATH this
  project has asked CMake to find a package configuration file provided by
  "console_bridge", but CMake did not find one.

Could not find a package configuration file provided by "console_bridge"
with any of the following names:

  console_bridgeConfig.cmake
  console_bridge-config.cmake

Add the installation prefix of "console_bridge" to CMAKE_PREFIX_PATH or set
"console_bridge_DIR" to a directory containing one of the above files. If
"console_bridge" provides a separate development package or SDK, be sure it
has been installed.

-- Configuring incomplete, errors occurred!
See also "/ros_ws/build/CMakeFiles/CMakeOutput.log".
See also "/ros_ws/build/CMakeFiles/CMakeError.log".
Invoking "cmake" failed
```

Complication: Have to be specified in multiple locations

```
cmake_minimum_required(VERSION 2.8.3)
project(libmavconn)

find_package(catkin REQUIRED)
+ find_package(console_bridge REQUIRED)
find_package(Boost REQUIRED COMPONENTS system)
...

catkin_package(
  INCLUDE_DIRS include
  LIBRARIES mavconn
+ DEPENDS Boost console_bridge mavlink
  CFG_EXTRAS libmavconn-extras.cmake
)

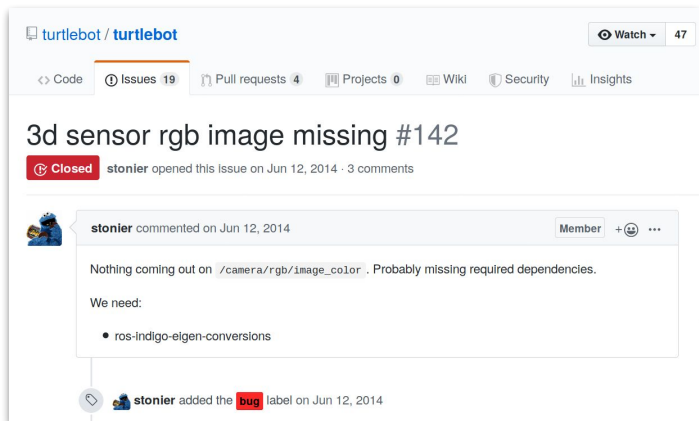
include_directories(
  include
  ${CMAKE_CURRENT_BINARY_DIR}/catkin_generated/include
  ${Boost_INCLUDE_DIRS}
  ${mavlink_INCLUDE_DIRS}
+ ${console_bridge_INCLUDE_DIRS}
)
...
target_link_libraries(mavconn
  ${Boost_LIBRARIES}
+ ${console_bridge_LIBRARIES}
)
```

```
<?xml version="1.0"?>
<package format="2">
  <name>libmavconn</name>
  <version>0.9.0</version>
  <description>
    MAVLink communication library.
    This library provide unified connection
    handling classes and URL to connection
    object mapper.
    ...
  </description>
  ...
  <buildtool_depend>catkin</buildtool_depend>
  <depend>boost</depend>
  <depend>mavlink</depend>
+ <depend>libconsole-bridge-dev</depend>
  <test_depend>gtest</test_depend>
  ...
</package>
```

2. Missing runtime dependencies!

- Shared objects / DLLs
- Python modules
- ROS nodes
- Configuration files
- ...

```
id: f01d952
title: No Image Coming From Camera
system: turtlebot
description: >
    A missing runtime dependency crashed the image processing node,
    causing no images to be received from the camera.
...
```



The screenshot shows a GitHub issue page for the repository 'turtlebot / turtlebot'. The issue is titled '3d sensor rgb image missing #142' and is marked as 'Closed'. It was opened by user 'stonier' on June 12, 2014, with 3 comments. A comment from 'stonier' on the same date states: 'Nothing coming out on /camera/rgb/image_color. Probably missing required dependencies. We need: • ros-indigo-elgen-conversions'. A 'bug' label was added to the issue on the same date.

```
[ INFO] [1402564815.530736554]: /camera/rgb/camera_info ->
/camera/rgb/camera_info
[ERROR] [1402564815.727176562]: Failed to load nodelet
[/camera/depth_metric_rect] of type
[depth_image_proc/convert_metric]: Failed to load library
/opt/ros/indigo/lib//libdepth_image_proc.so.
Make sure that you are calling the PLUGINLIB_EXPORT_CLASS macro in the
library code, and that
names are consistent between this macro and your XML. Error string: Could not
load library (Poco exception = libeigen_conversions.so: cannot open shared
object file: No such file or directory)
[FATAL] [1402564815.727410623]: Service call failed!
```

3. Dangerous defaults

```
# mavros/mavros/scripts/mavcmd

def do_long(args):
    try:
        ret = command.long(
            broadcast=args.broadcast,
            command=args.command,
            confirmation=int(args.confirmation),
            param1=args.param1,
            param2=args.param2,
            param3=args.param3,
            param4=args.param4,
            param5=args.param5,
            param6=args.param6,
            + param7=args.param7)
    except rospy.ServiceException as ex:
        fault(ex)
    _check_ret(args, ret)
```

id: [c6791f0](#)

title: Missing parameter in ROS service call

system: mavros

description: >

The "do_long" function in the "mavcmd" script accepts an object that describes a MAVLink command. Command objects are supplied with seven arguments, which may be used or ignored by the command handler depending on the type of command. The "do_long" function dispatches this command object to the "mavros.command.long" function, which forwards the command to MAVLink. Rather than accepting positional arguments, the "mavros.command.long" accepts all of its arguments as keywords. Missing keywords are assumed to be irrelevant and substituted by a nominal placeholder value.

The buggy implementation of this function fails to pass along the seventh argument during the call to "mavros.command.long", causing incorrect information to be forwarded to MAVLink. This bug will only manifest for commands which utilise all seven arguments.

Also see: [#e1a8005](#); [#ff581a0](#)

4. String identifiers: Silent but dangerous

```
<!--  
  Auto-docking + keyop configuration for working with the default kobuki  
  launcher (minimal.launch).  
-->  
<launch>  
  <node pkg="nodelet"  
    type="nodelet"  
    name="cmd_vel_mux"  
    args="load cmd_vel_mux/CmdVelMuxNodelet  
mobile_base_nodelet_manager">  
    <param name="yaml_cfg_file"  
      value="$(find kobuki_auto_docking)/param/cmd_vel_mux.yaml"/>  
-   <remap from="cmd_vel_mux/mux_cmd_vel"  
+   <remap from="cmd_vel_mux/output"  
      to="mobile_base/commands/velocity"/>  
  </node>  
  ...  
</launch>
```

```
id: 35682ec  
system: kobuki  
title: Wrong topic remapping makes robot ignore velocity commands  
description: >  
  ROS allows resource names to be remapped. Topics, in particular,  
  are resources that can be remapped. When topic "A" is remapped to  
  "B" and a node tries to publish/subscribe on "A", it will  
  (automatically) publish/subscribe to "B" instead.  
  Remappings are commonly defined in launch files. Launch files are  
  special XML files used by the tool 'roslaunch' to initialize  
  several nodes, parameters and remappings at once.  
  In this particular bug, a node (cmd_vel_mux) publishes velocity  
  messages that are received by the driver node (mobile_base). A  
  launch file defined a remapping from "cmd_vel_mux/mux_cmd_vel" to  
  "mobile_base/commands/velocity", but the cmd_vel_mux node is, in  
  fact, publishing on "cmd_vel_mux/output". Since the node is  
  publishing on a different topic, the remapping has no effect, and  
  so the two expected topic names are not connected. The end result  
  is that the published messages never reach their destination,  
  i.e., the robot does not move at all.  
  This issue is very specific to ROS, as it is located and solved  
  in launch.xml files, but the issue of a runtime configuration of  
  a distributed system is general, in principle.
```

Also see [43705f7](#), [b18f559](#), [fbe70c7](#), [263650d](#)

5. Namespace confusion — the defaults strike back!

```
class VisualizationPlugin : public MavRosPlugin {  
public:  
    VisualizationPlugin() : viz_nh("~visualization"), ... { };  
  
    void initialize(...) {  
        ...  
-     viz_nh.param<std::string>("visualization/fixed_frame_id",  
+     viz_nh.param<std::string>("fixed_frame_id",  
                                fixed_frame_id, "local_origin");  
        ...  
    }  
}
```

`/visualization/visualization/fixed_frame_id`
`/visualization/fixed_frame_id`

Also see [#1518978](#), [#9ffffca](#), [#a482f82](#)

id: [84264f0](#)

title: Incorrect use of namespacing in ROS parameter names

description: >

All ROS parameter interactions within

"`mavros_extras/src/plugins/visualization.cpp`" incorrectly prefix the name of parameters with a "visualization" namespace. This namespace is implicit, and so the fully qualified parameter that result are incorrect. e.g., "`visualization/fixed_frame_id`" becomes "`/visualization/visualization/fixed_frame_id`" when fully qualified.

classification: "Incorrect use of ROS namespacing (no CWE)"

...

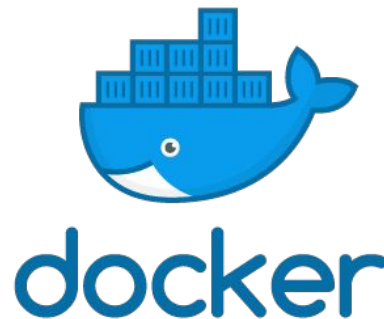
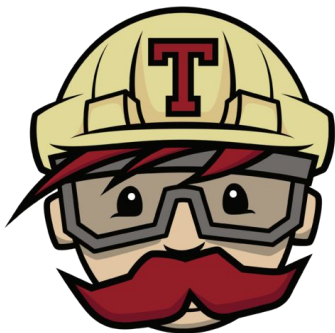
How can I build better software today?



Make your life easier with continuous integration

- Identify unspecified build-time dependencies!
- Use smoke tests to find missing run-time dependencies
- Use test automation to identify regressions
- Incorporate other tools into CI pipeline...

build passing



Use linters to statically catch errors before run-time



- roslint
- catkin_lint
- ament_lint (ROS2)
- ...



Use sanitizers to catch runtime errors earlier

GCC and Clang offer sanitizers that perform various safety checks on your code:

- Buffer overflows
- Memory leaks
- Initialization order bugs
- Data races
- Deadlocks
- Uninitialized memory
- Undefined behaviour
- ...

```
$ catkin_make \  
-DCMAKE_CXX_FLAGS="-fsanitize=address"
```



Use static type checking in Python (PEP 484)

Roughly one in twelve bugs was a type error!



```
from typing import Iterator

def fib(n: int) -> Iterator[int]:
    a, b = 0, 1
    while a < n:
        yield a
        a, b = b, a + b
```

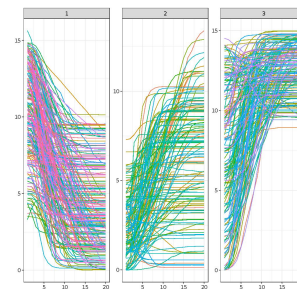
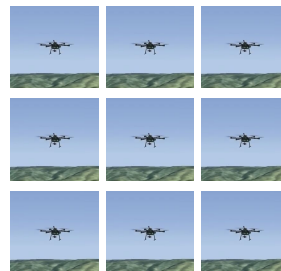
```
def is_palindrome(s):
    # type: (str) -> bool
    return s == s[::-1]
```


What's next?



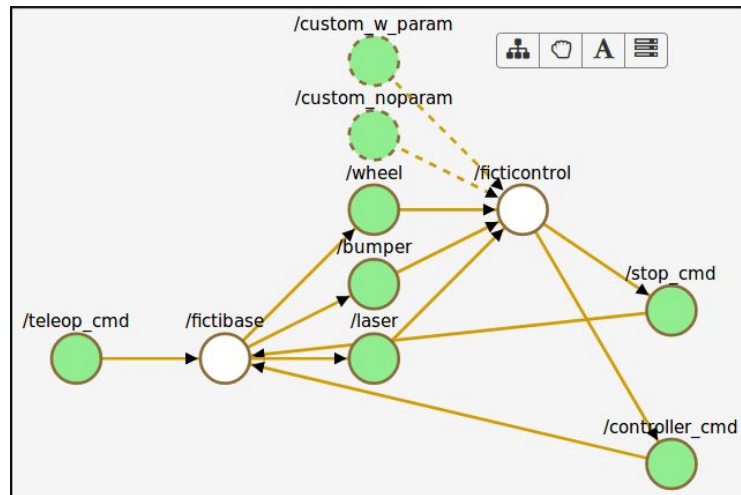
Mithra: Oracle learning for simulation-based testing

- Operates on logged data (e.g., ROS bag files)
- Uses multivariate time series clustering to identify intended system behaviours
- Uses anomaly detection to identify erroneous executions



HAROS: A static analysis framework for ROS apps

- Understands package and workspace structure
- Unifies reports from various tools (e.g., metrics, linters)
- Able to extract the ROS Graph from code (C++ as of now; Python in the next release)
- Produces interactive reports



<https://github.com/git-afsantos/haros/>

Shout out to Phriky-Units

Detects physical unit inconsistencies for C++ **without the need for annotations.**

```
189 float computeDistance(geometry_msgs::Pose goal, geometry_msgs::Pose current)
190 {
191     float dist = sqrt((goal.position.x - current.position.x)*(goal.position.x - current.position.x)
+ (goal.position.y - current.position.y) + (goal.position.y - current.position.y)
+ (goal.position.z - current.position.z) + (goal.position.z - current.position.z));
```

.....meters-squared

.....meters

<https://github.com/unl-nimbus-lab/phriky-units>

188 ROS bugs later: Where do we go from here?

Christopher Timperley

ROSCon 2019, Macau



ROBUST: ROS Bug Study



```
$ rosinstall_generator_time_machine
```



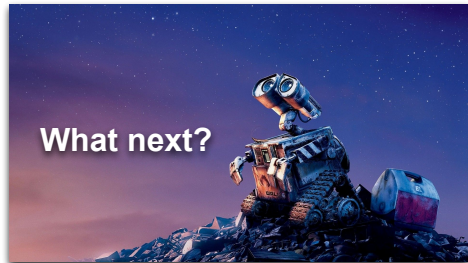
What kinds of bugs do ROS
developers encounter?



How can I build
better software
today?



What next?



github.com/robust-rosin/robust