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

Christopher Timperley

ROSCon 2019, Macau
An unlikely beginning...
An unlikely beginning...
What kinds of bugs do ROS developers encounter?

Understanding ROS bugs

Tools, Tips, and Techniques
Meet the ROBUST team!
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 266 bugs across 8 ROS packages

<table>
<thead>
<tr>
<th>Subject</th>
<th># Bugs</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAVROS</td>
<td>57</td>
</tr>
<tr>
<td>Kobuki</td>
<td>40</td>
</tr>
<tr>
<td>Universal Robot</td>
<td>25</td>
</tr>
<tr>
<td>Motoman</td>
<td>22</td>
</tr>
<tr>
<td>TurtleBot</td>
<td>12</td>
</tr>
<tr>
<td>Care-O-Bot</td>
<td>11</td>
</tr>
<tr>
<td>geometry2</td>
<td>36</td>
</tr>
<tr>
<td>Confidential</td>
<td>10</td>
</tr>
</tbody>
</table>
We analysed 1790 GitHub issues and pull requests

<table>
<thead>
<tr>
<th>Subject</th>
<th># Bugs</th>
<th># Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAVROS</td>
<td>57</td>
<td>325</td>
</tr>
<tr>
<td>Kobuki</td>
<td>40</td>
<td>623</td>
</tr>
<tr>
<td>Universal Robot</td>
<td>25</td>
<td>158</td>
</tr>
<tr>
<td>Motoman</td>
<td>22</td>
<td>78</td>
</tr>
<tr>
<td>TurtleBot</td>
<td>12</td>
<td>170</td>
</tr>
<tr>
<td>Care-O-Bot</td>
<td>11</td>
<td>182</td>
</tr>
<tr>
<td>geometry2</td>
<td>42</td>
<td>254</td>
</tr>
<tr>
<td>Confidential</td>
<td>10</td>
<td>N/A</td>
</tr>
</tbody>
</table>
We forensically described each bug
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

What kinds of bugs do ROS developers encounter?
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 namespaces
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
Function call with incorrect arguments
Uncatched exceptions
Information leak
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 namespaces
Incorrect mathematical operations
Lack of synchronisation leads to race condition
Bad name remappings
Incorrect constant values
Hardcoded values lead to silent errors
Encoding errors
More than a few...
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>
  ...
</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.
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>
  ...
</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.

```
[ 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

```python
# 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" to="mobile_base/commands/velocity"/>
  </node>
  ...
</launch>

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");
            ...
        }
};

Also see #1518978, #9fffca, #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...
Use linters to statically catch errors before run-time

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

ROS
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"
```

https://github.com/google/sanitizers
Use static type checking in Python (PEP 484)

Roughly one in twelve bugs was a type error!

```python
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**.

```cpp
float computeDistance(geometry_msgs::Pose goal, geometry_msgs::Pose current) {
    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));
}
```

[https://github.com/unl-nimbus-lab/phriky-units](https://github.com/unl-nimbus-lab/phriky-units)
188 ROS bugs later: Where do we go from here?
Christopher Timperley
ROSCon 2019, Matalu

ROBUST: ROS Bug Study

What kinds of bugs do ROS developers encounter?

How can I build better software today?

What next?

$ rosinstall_generator_time_machine

github.com/robust-rosin/robust