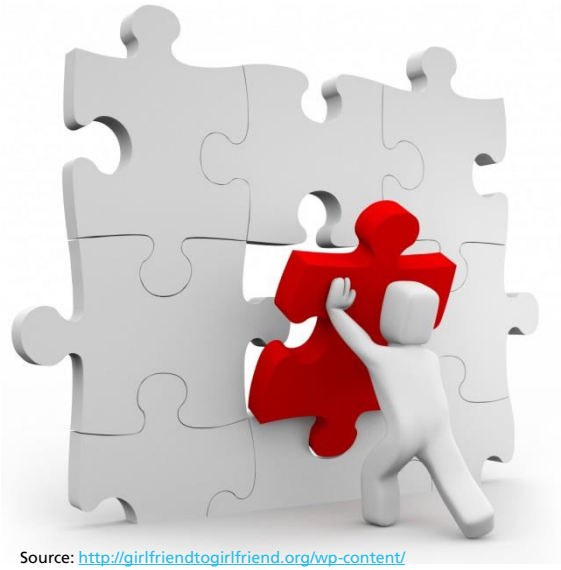
EFFICIENT SYSTEM INTEGRATION WITH AN AUTOMATED TESTING AND BENCHMARKING FRAMEWORK

Florian Weißhardt

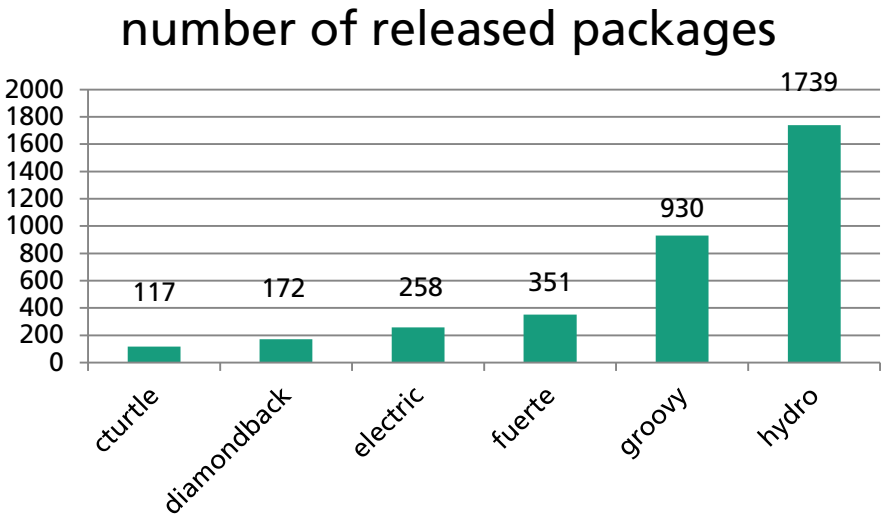


Source: http://girlfriendtogirlfriend.org/wp-content/uploads/2014/12/RAD_Puzzle-Photo.jpg

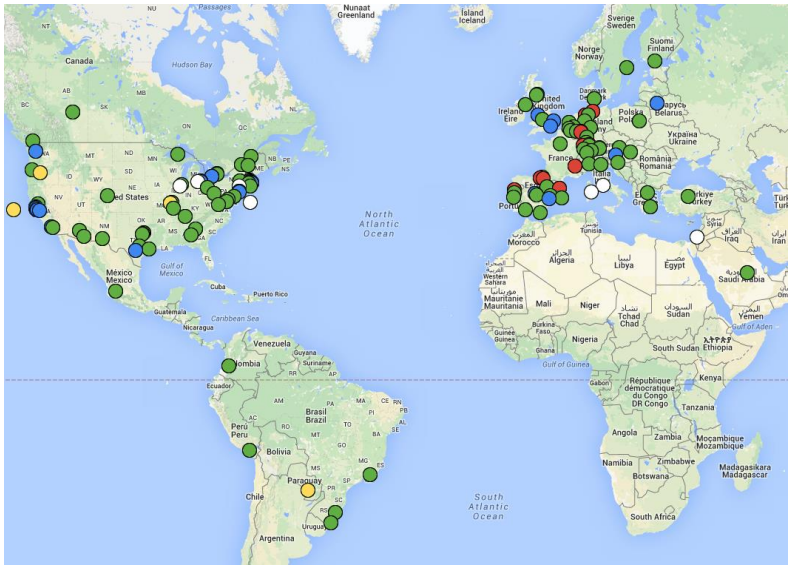
Fraunhofer IPA

Distributed Open Source Development within ROS

- The Robot Operating System ROS
  - Developed from a world wide Open Source community
  - Heterogeneous developers (student/hobbyist vs. professional)
  - Different points of interests (developer vs. consumer)
- Wide distribution of ROS: **annual growth between 36% and 165%** over the last 5 years



Source: ROS metrics reports http://wiki.ros.org/Metrics



Source: ROS Users Map http://metrorobots.com/rosmap.html

Fraunhofer IPA

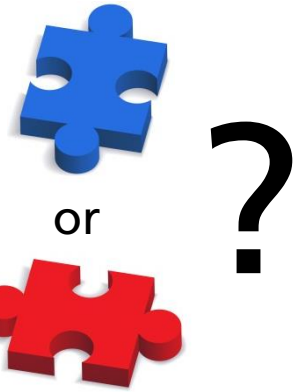
Distributed Open Source Development within ROS

- Information about a ROS package is spread over
  - ROS Wiki (package.xml and user specific documentation)
  - Source Code and Doxygen
  - Expert knowledge (direct contact, mailing lists and answers.ros.org)

- Quality measures for a ROS package
  - Compiling and static code analysis (OSRF buildfarm)
  - Unit testing on code level, but generally very poor test coverage
  - ROS-I rating (wiki)
  - Level of maintenance and activity (github and rosindex)
  - **No objective quality measures for runtime behavior**

Problem

- How to find out which component performs best…
  - … with other components?
  - … with my robot?
  - … in my environment?
  - … in my infrastructure?



Source: http://www.sharonostalecki.com/wp-content/uploads/2012/08/HiRes-puzzle-pieces.jpg

Source: http://girlfriendtogirlfriend.org/wp-content/uploads/2014/12/RAD_Puzzle-Photo.jpg

- The performance of a ROS system is influenced by

	Example
Composition of components	4
Component behavior & implementation	3
Component configuration (parameters)	9
Environment conditions	4
Available resources & deployment configuration	3
	=====

- Number of test cases ~ O(n!) > 1000

Solution

Need for quality meassures for runtime behaviour!

Need for objective testing!

Automated testing and benchmarking framework which

generates objective **performance indicators** through common metrics.

Testing and Benchmarking Framework

- Offer **infrastructure** for automating tests
  - Continuous integration service
  - Hardware- and Simulation-in-the-loop testing

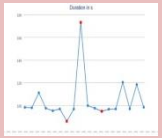
- Offer **generic building blocks** for specifying new tests
  - System level (CPU, RAM, IO, Network, …)
  - Node level (Update frequency, publish frequency, …)
  - Behaviour specific (TF comparison, jerk execution profile, …)

- Offer **benchmarking tests** for commonly used ROS interfaces (e.g. MoveBase, Object Detection, MoveGroup)
  - Based on specific metrics for each interface
  - E.g. execution time, accuracy, path length, …

Architecture

Automated test and evaluation framework based on Jenkins CI

Evaluation and Analysis



Recorded data

Test data

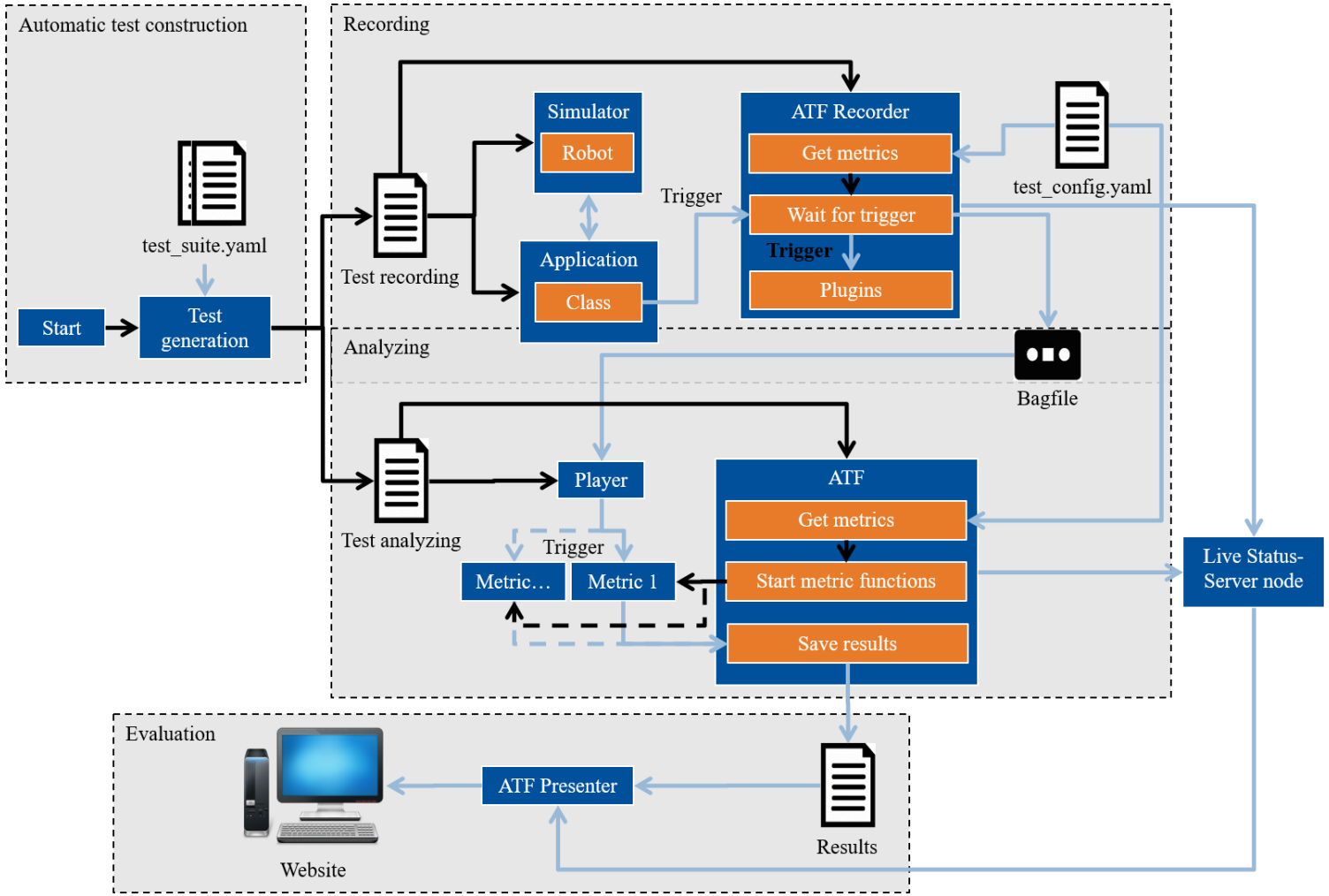
Simulation-in-the-loop

Hardware-in-the-loop

Checkout & Build & Package

Code Repositories

Architecture



Fraunhofer IPA

ATF metrics

- Available metrics, see https://github.com/ipa-fmw/atf

Metric	Description	Unit	Mode (span, snap, min/max)
time	The time metric measures the elapsed time.	[sec]	span
path_length	The path_length metric measures the cartesian path (distance integrated over time) of a TF frame with respect to another frame.	[m]	span
publish_rate	The publish_rate metric measures the publising rate of a topic	[1/sec]	span
api	The api metric checks if an interfaces (nodes, publishers, subscribers, service servers, action servers) matches its specification.	[bool]	snap

- Further metrics (in development)

ressources	The ressources metric measures the ressource consumption of a node on the operating system level (CPU, RAM, IO).	[%], [MB], [MB/sec]	snap
path_velocity	The path_velocity metric measures the cartesian velocity (distance differntiated over time) of a TF frame with respect to another frame.	[m/sec]	span
distance	The distance metric measures the cartesian distance between two TF frames.	[m]	snap, min/max
obstacle_distance	The obstacle_distance metric measures the distance between two meshes	[m]	snap, min/max
message_match	The message_match metric checks if a message content matches its desired content.	[bool]	snap

Example ATF package

- Configuration separated into robot, environment and application specific configuration
- Automatic test case generation based on test suites

```
1  | -- CMakeLists.txt
2  | -- config
3  |     | -- robot_envs
4  |     |     | -- env1.yaml
5  |     | -- robots
6  |     |     | -- robot1.yaml
7  |     |     | -- robot2.yaml
8  |     | -- test_configs
9  |     |     | -- test1.yaml
10 |     |     | -- test2.yaml
11 |     | -- test_generation_config.yaml
12 |     | -- test_suites.yaml
13 | -- launch
14 |     | -- application.launch
15 | -- package.xml
16 | -- scripts
17       | -- application.py
```

Defining a test suite

- Easy configuration for test case variation (from combination of robot, environment and application specific configuration)

```yaml
# this will generate 2 (=1*2*1) test cases and 20 Tests in total (=2*10)
testsuite_1:
  test_configs:
    - test1
  robots:
    - robot1
    - robot2
  robot_envs:
    - env1
  repetitions: 10

# this will generate 12 (=2*3*2) test cases and 120 Tests in total (=12*10)
testsuite_2:
  test_configs:
    - test1
    - test2
  robots:
    - robot1
    - robot2
    - robot3
  robot_envs:
    - env1
    - env2
  repetitions: 10
```

Test blocks

- Defining metrics to be used within test block
- Optional groundtruth data for automatic evaluation

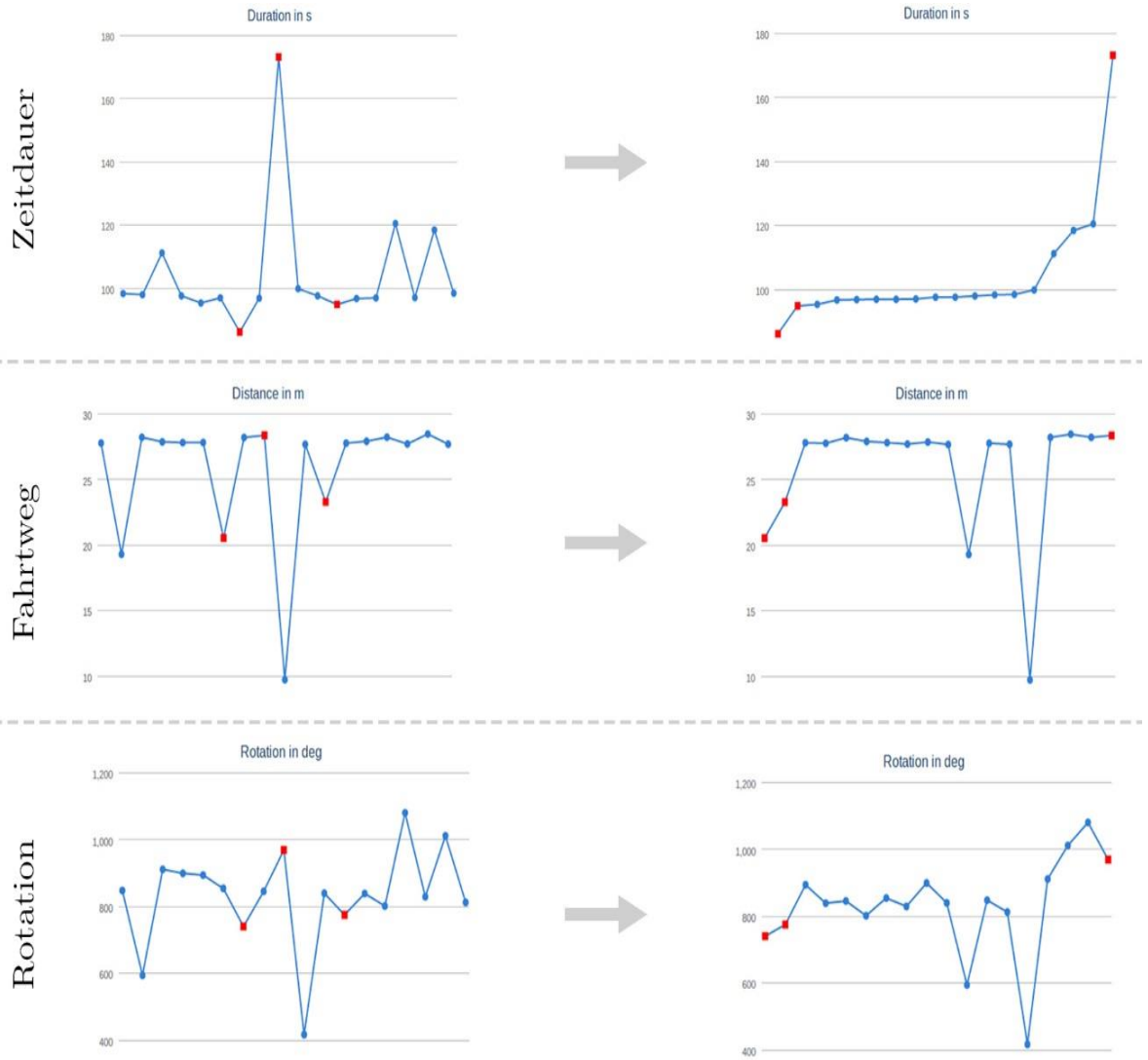
```
1  testblock_1:
2    time:
3      - groundtruth: 3
4        groundtruth_epsilon: 0.5
5    path_length:
6      - root_frame: link1
7        measured_frame: link2
8        groundtruth: 12.00
9        groundtruth_epsilon: 1.0
10   publish_rate:
11     - topic: topic1
12       groundtruth: 10
13       groundtruth_epsilon: 1
```

ATF application using test blocks

- Available python API for easy integration into user application

```python
from atf_core import ATF

class Application:
    def __init__(self):
        # Initialize the ATF class
        self.atf = ATF()

    def execute(self):
        # You can call start/pause/purge/stop for each testblock
        # during the execution of your app

        self.atf.start("testblock_all")
        self.atf.start("testblock_1")

        # Do something

        self.atf.stop("testblock_1")
        self.atf.start("testblock_2")

        # Do something else

        self.atf.stop("testblock_2")
        self.atf.stop("testblock_all")

        # Finally we'll have to call shutdown() to tell
        # the ATF to stop all recordings and wrap up
        self.atf.shutdown()
```

Example Navigation: Benchmarking Results



Fraunhofer IPA

Example Navigation: Hardware vs. Simulation

Filter Criteria

From: any

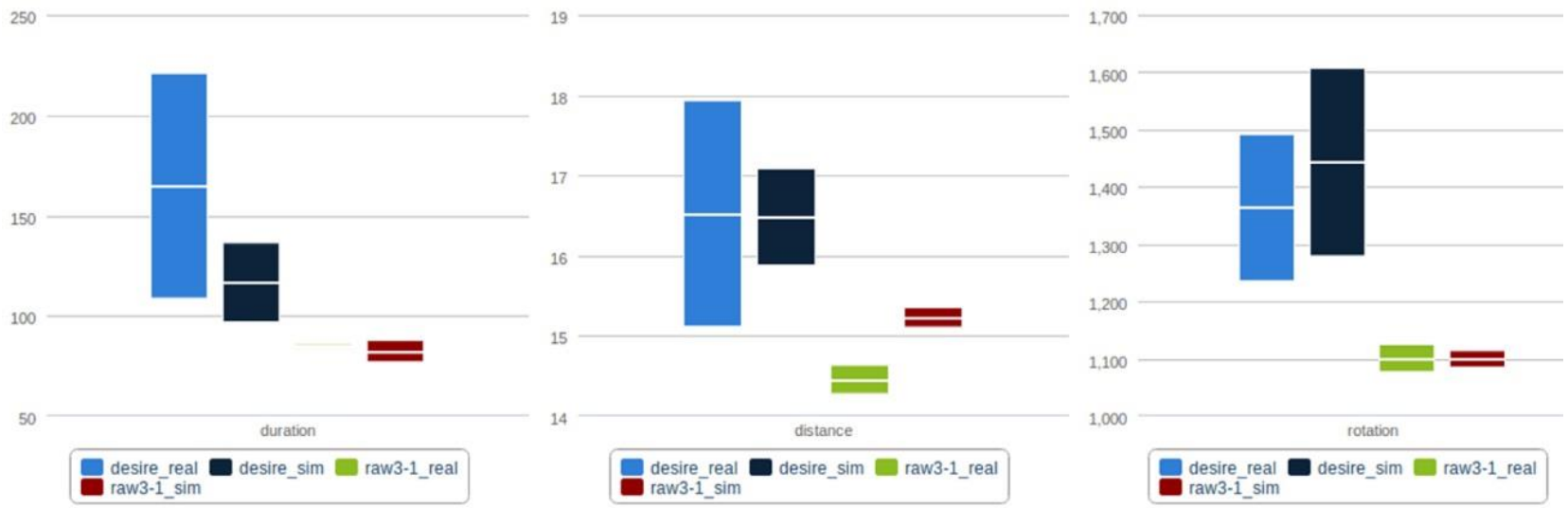
To: any

Last n: any Tests

&& ||

	#	#ERR	#ABRTD	#FLD	#MISS	#TO	#Col	Roboter	Navigation	Scenario	Duration	Distance	Rotation
☐	18	3	3	0	0	0	0	cob3-3	2dnav_ros_tr	ipa-apmt	101.18	26.12	830.56
☐	8	6	6	0	0	0	0	desire_sim	2dnav_ros_tr	ipa-apmt	134.01	30.6	1061.53
☐	15	4	3	0	1	0	0	cob3-3	2dnav_ros_dwa	room	94.7	13.88	1041.41
☐	10	0	0	0	0	0	0	cob3-3	2dnav_ipa_extloc	room	163.81	15.8	745.57
☑	26	16	13	1	1	1	0	desire_real	2dnav_ros_dwa	room	164.83	16.52	1363.66
☑	25	17	15	0	1	1	0	raw3-1	2dnav_ros_dwa	room	82.37	15.23	1099.95
☐	17	0	0	0	0	0	0	raw3-1	2dnav_ros_dwa	ipa-apmt	101.8	29.63	638.95
☐	17	2	2	0	0	0	0	raw3-1	2dnav_ros_tr	ipa-apmt	100.07	29.26	662.46

Application Developer | Component Developer



Fraunhofer IPA

Summary

- Goal: Enhance **transparency for component quality** in distributed Open Source development
  - Make runtime performance of a ROS system transparent to the user
  - Help the ROS user to select the best component out of the multitude of ROS components

- Solution: **Automated testing and benchmarking framework** which generates objective **performance indicators** through common metrics
  - Testing infrastructure
  - Generic building blocks
  - Benchmarking tests
  - Hardware- and Simulation-in-the-loop testing

- Source Code and examples on github: https://github.com/ipa-fmw/atf