Johannes Mey[1], Ariel Podlubne[1,2], René Schöne[1], Paul Gottschaldt[1,2], Diana Göhringer[1,2], Uwe Aßmann[1,2]

[1]Technische Universität Dresden
[2]Centre for Tactile Internet with Human-in-the-Loop (CeTI)

Systematic Testing of a ROS Interface Specification Backend

6th International Workshop on Robotics Software Engineering (RoSE'24)
Lisbon, April 15th 2024

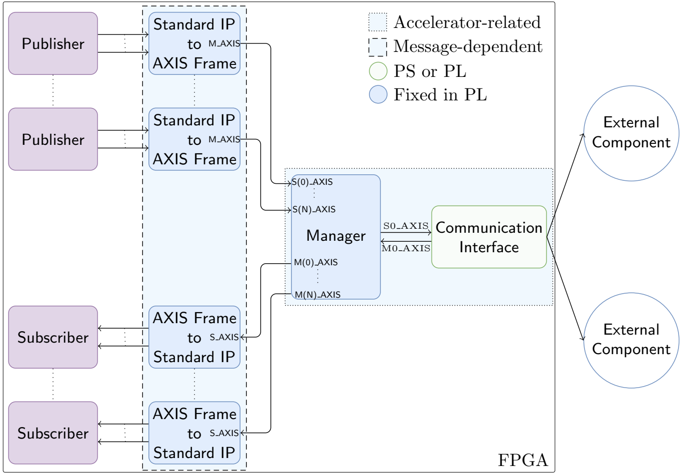
The System Under Test: FIRM [Pod+21]

“FIRM”:

- FPGA (VHDL)
  ROS 1 and ROS 2 Middleware

Goal:

- Receive ROS messages on the hardware (PL) bypassing the CPU (PS)



[Pod+21] Ariel Podlubne et al. “Model-Based Approach for Automatic Generation of Hardware Architectures for Robotics”. In: *IEEE Access* 9 (2021). ISSN: 2169-3536

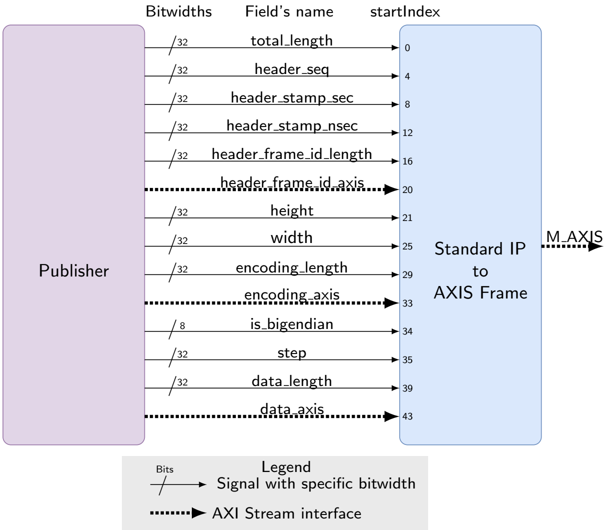
The System Under Test: FIRM [Pod+21]

“FIRM”:

- FPGA (VHDL)
  ROS 1 and ROS 2 Middleware

Goal:

- Receive ROS messages
  on the hardware (PL)
  bypassing the CPU (PS)



[Pod+21] Ariel Podlubne et al. “Model-Based Approach for Automatic Generation of Hardware Architectures for Robotics”. In: *IEEE Access* 9 (2021). ISSN: 2169-3536

Creating a ROS Middleware for FPGAs

ROS Middleware

- Communication components in **library**
- **Generated bindings** for each ROS message type

ROS Message Types

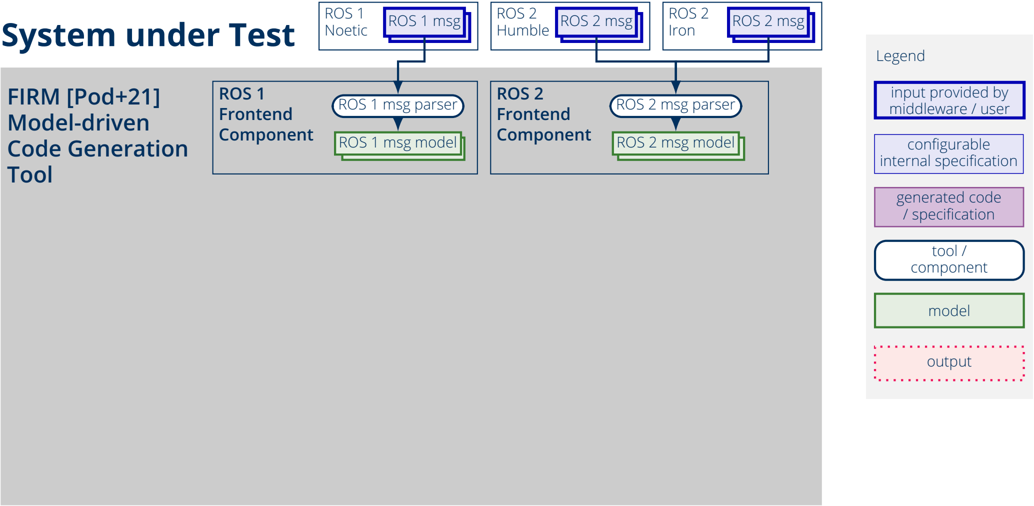
- Custom format in ROS1
- Mapped to OMG IDL in ROS2

```
std_msgs/Header header
    uint32 seq
    time stamp            # nested message
    string frame_id
uint32 height
uint32 width
string encoding           # unconstrained size
uint8 is_bigendian
uint32 step
uint8[] data
```

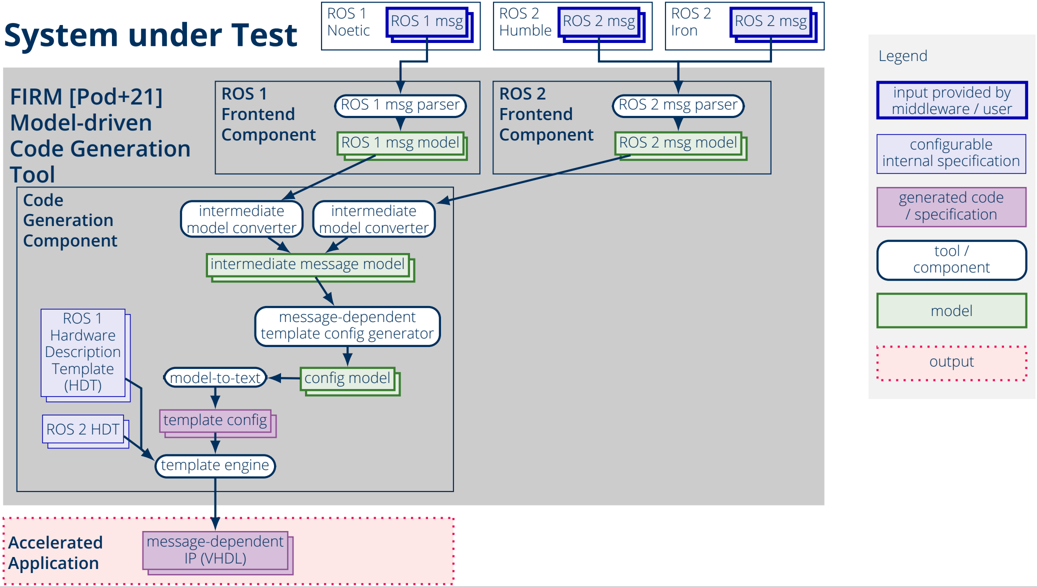
Challenges

- Support **all** ROS 1 and 2 versions
- Support **multiple** FPGA **vendors**/VHDL **dialects**
- ROS message **complexity**
- Testing on **FPGA-hardware**
- Distributed **skills**

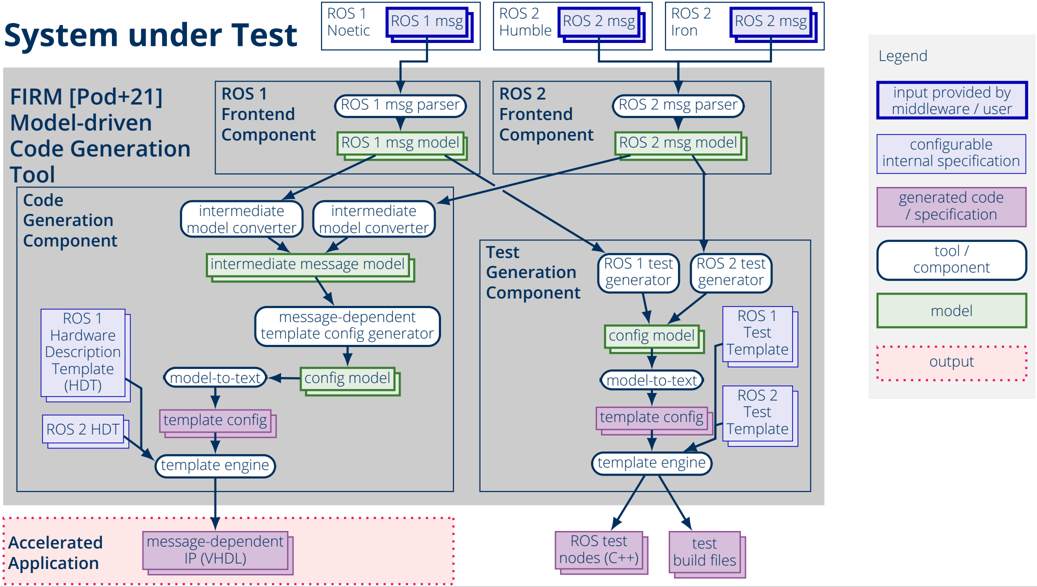
System under Test



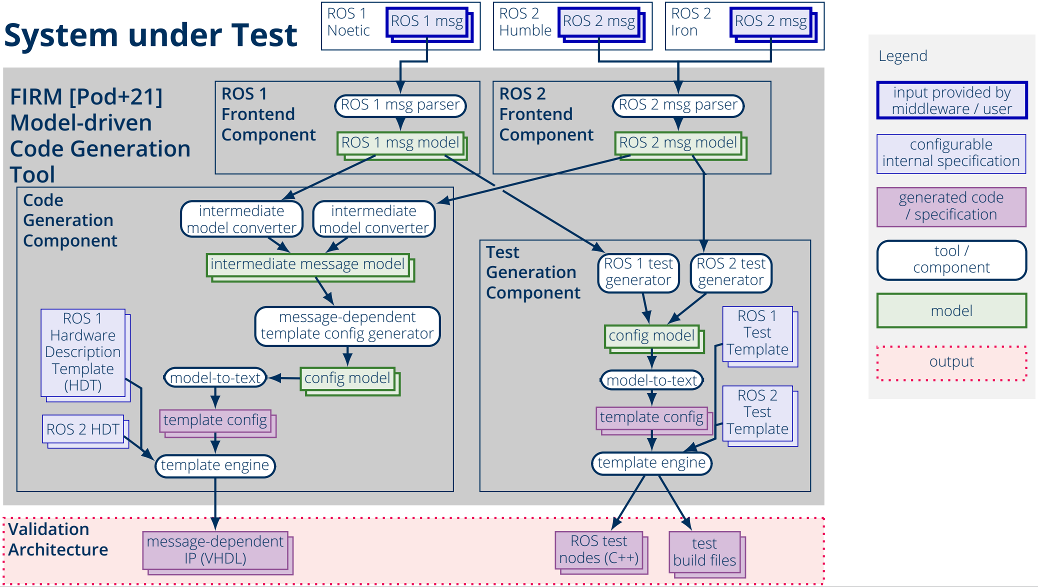
System under Test



System under Test



System under Test



Test Stages

Frontend Tests (TS1)

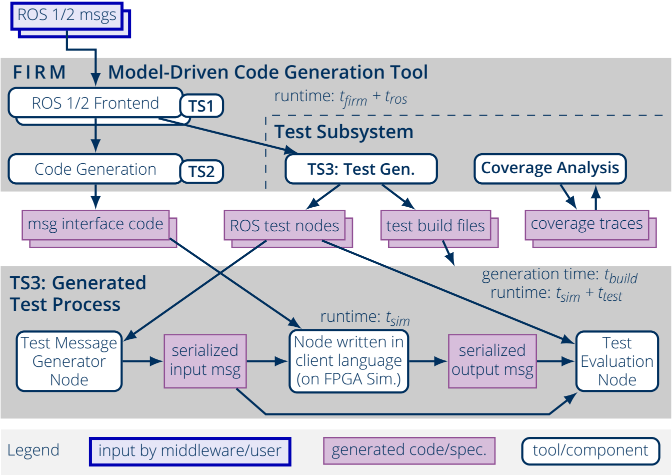
- ROS integration
- Parser

Code Generation tests (TS2)

- Regression tests

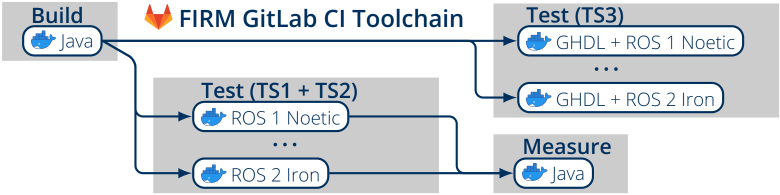
Runtime tests (TS3)

- Generate messages
- Pipe through FPGA (sim)
- Compare input/output
- **Only frontend shared with FIRM**



Execution

- Dockerized Gitlab CI Pipeline
- Automatic ROS1/2 switch based on ROS system variable
  → add new ROS version = add new base image



Strategies / Insights / Lessons Learned

- **Specification**
- Test in Stages
- **Use Analysis**
- **Manage Test Effort**
- **Assess Coverage**

Specification

ROS 1

- Informal specification
- Assumption:
  "It's a ROS message if it works in Python and C++"

Specification

ROS 1

- Informal specification
- Assumption:
  "It's a ROS message if it works in Python and C++"

ROS 2

- DDS → based on OMG IDL
- Message format itself still informal
- Transformation ROS to IDL informal

Specification

ROS 1

- Informal specification
- Assumption:
  “It’s a ROS message if it works in Python and C++”

ROS 2

- DDS → based on OMG IDL
- Message format itself still informal
- Transformation ROS to IDL informal

→ Is testing all existing ROS messages enough?

Specification

ROS 1

- Informal specification
- Assumption:
  "It's a ROS message if it works in Python and C++"

ROS 2

- DDS → based on OMG IDL
- Message format itself still informal
- Transformation ROS to IDL informal

→ Is testing all existing ROS messages enough?

Not in paper: Combination of fuzzing and Controllable Combinatorial Coverage.

Structure of ROS Messages: Analysis and Metrics

Implementation using **Reference Attribute Grammars** [Hed00] with **JastAdd** [EH07]

→ **Analysis capabilities**

Properties

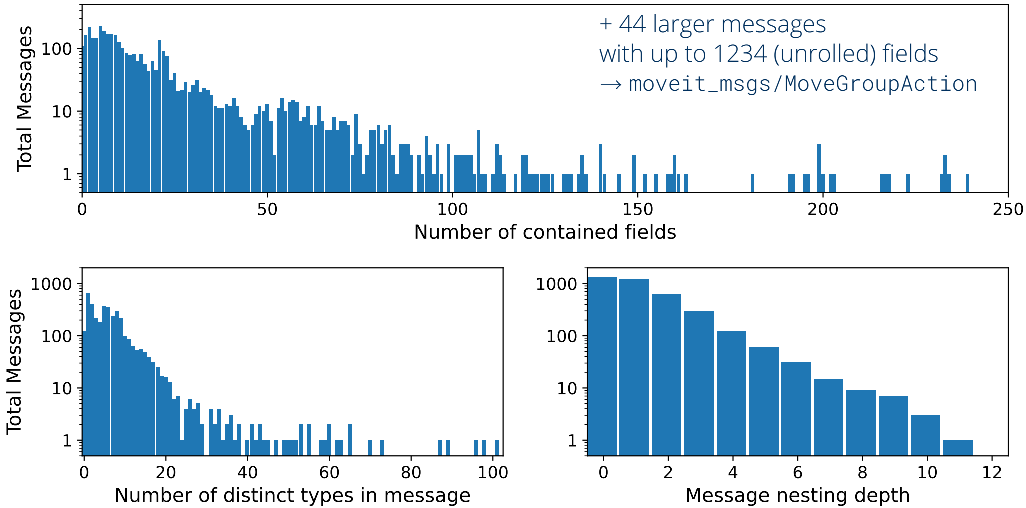
- containsSubmessages
- containsUnconstrainedSubmessages
- containsUnconstrainedVariables
- containsStrings
- containsConstants
- isPartOfAction
- ...

Metrics

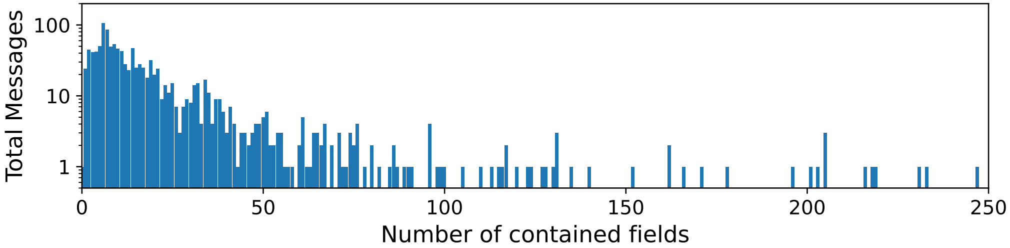
- nestingDepth
- numberOfDataFields
- distinctTypes
- distictPrimitiveTypes
- distinctMessageTypes
- ...

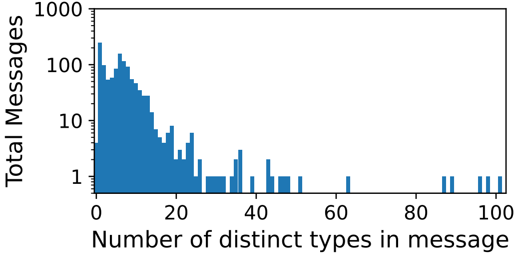
[Hed00] Görel Hedin. “Reference attributed grammars”. In: *Informatica (Slovenia)* 24.3 (2000), pp. 301–317
[EH07] Torbjörn Ekman and Görel Hedin. “The JastAdd system — modular extensible compiler construction”. en. In: *Science of Computer Programming*. Special issue on Experimental Software and Toolkits 69.1 (2007), pp. 14–26. ISSN: 0167-6423

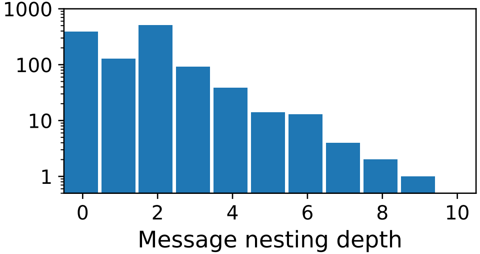
Distribution of ROS Messages in ROS1 Noetic



Distribution of ROS Messages in ROS2 Humble

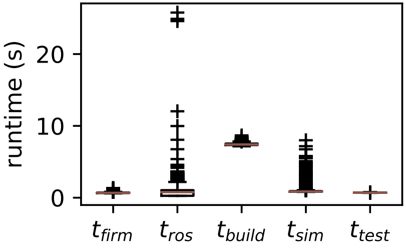






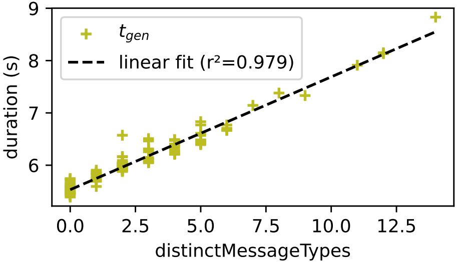
Test Runtime Analysis

Runtime of Tests



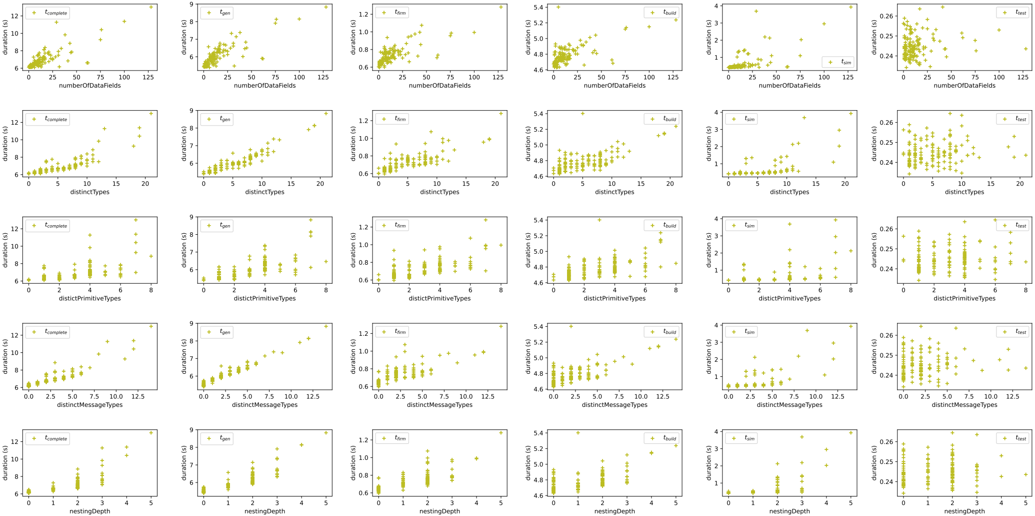
- Getting ROS message **expensive**
- Constant build time

Correlation to Properties

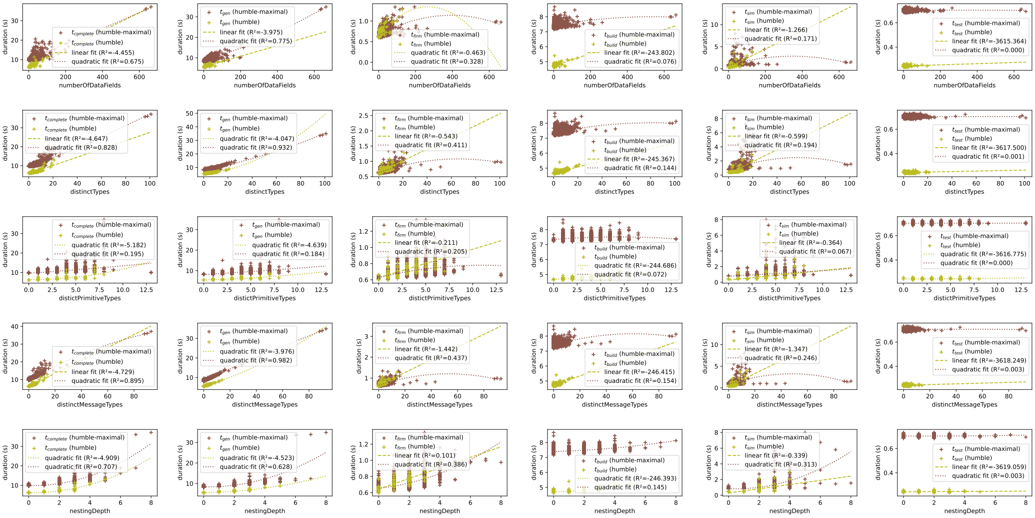


- ROS version *(ROS2 Humble)*
- Test phase *(t_{gen})*
- Property
  (Number of contained distinct message types)

Scatterplot: Metrics x Time in Phase



Scatterplot: Standard Packages vs All Packages



Coverage

Problem: Coverage of elements in templates

1. Assign a number to each text fragment and create a lookup table

#	Template File	Pos.	Stack	Content
0	`template1.mustache`	(1, 1)		`""`
1	`template1.mustache`	(1,13)	#msg	`"\n"`
2	`template1.mustache`	(2,12)	#msg>#fields	`"\n"`
3	`template1.mustache`	(3,12)	#msg>#fields>#simple	`"\n"`
4	`template1.mustache`	(4,10)	#msg>#fields>#simple>#axis	`"\n{{name}}_tready_in when s_counter"`
5	`template1.mustache`	(5,53)	#msg>#fields>#simple>#axis>#currentMsg	`"_{{currentMessage}}"`
6	`template1.mustache`	(5,91)	#msg>#fields>#simple>#axis	`"={{index_tdata}} else\n"`
7	`template1.mustache`	(6,10)	#msg>#fields>#simple	`"\n"`

2. Create copy of templates replacing all fragments with just the number

```
[0]{{\#message}}[1]{{#fields}}[2]{{#simple}}[3]{{#axis}}[4]{{#currentMessage}}[5]{{/currentMessage}}[6]{{/axis}}[7]
```

3. Run the test suite, obtaining number sequences
4. Aggregate all numbers, thus finding missing fragment indices
5. Identify dead code using the lookup table

Conclusion

Summary

- o ~30k tests
- \+ High confidence in FIRM quality
- \+ Test systems allow expert collaboration
- \+ Data about ROS message landscape
- – Blocking factor specification
- – No good minimal test set yet

Opportunities and Next Steps

- Apply fuzzing and Controllable Combinatorial Coverage to generate test set
- ROS Message → OMG IDL
- Applicable to any middleware backend