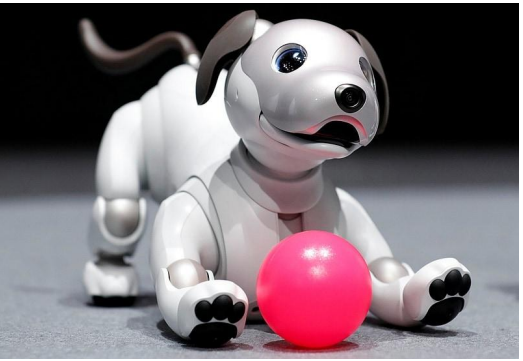
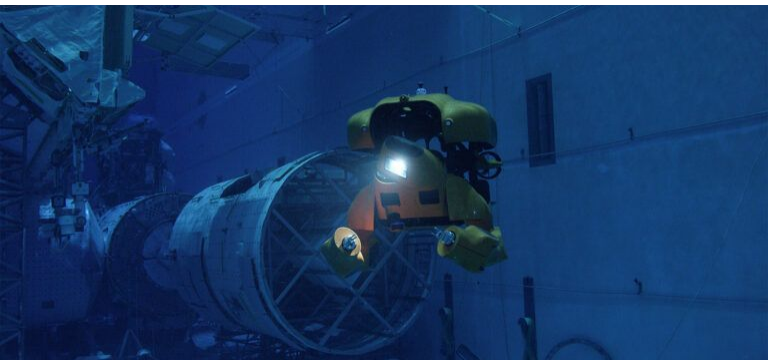
Tools and Processes for Improving the Certifiability of ROS 2

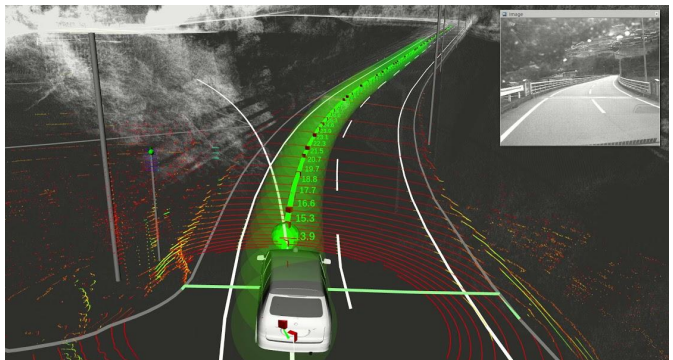
Michael Jeronimo and Geoffrey Biggs

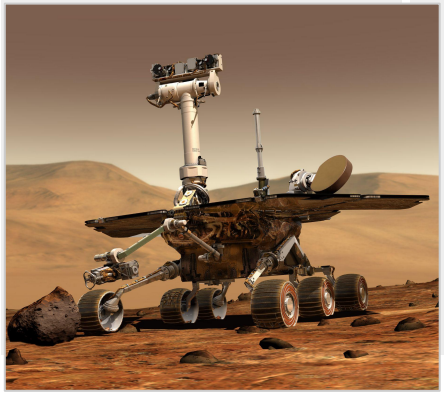
Open Robotics

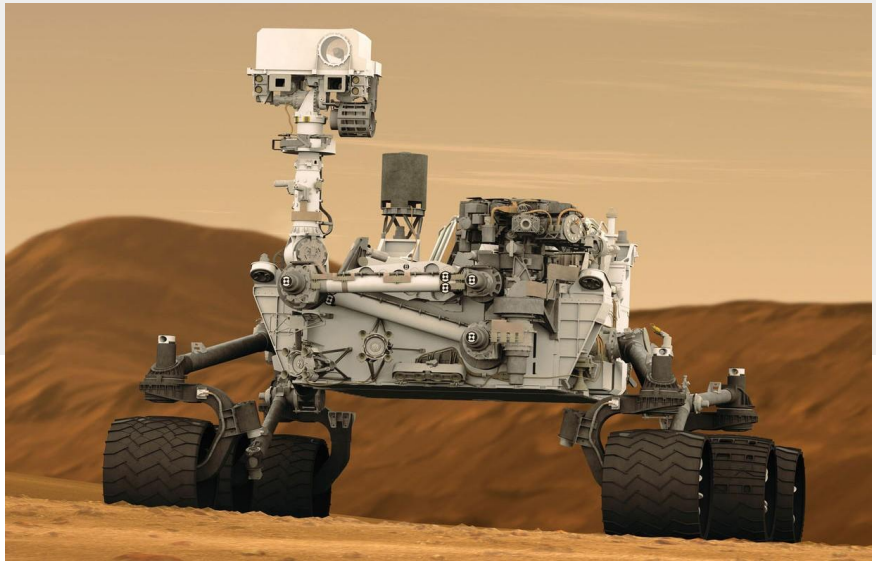


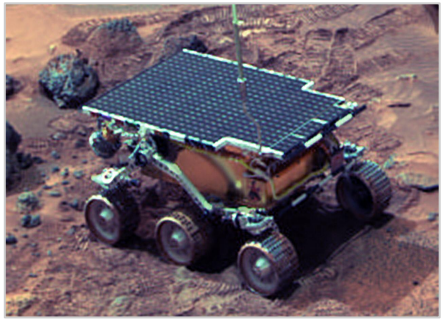
(ROS-based) Robots are everywhere on Earth



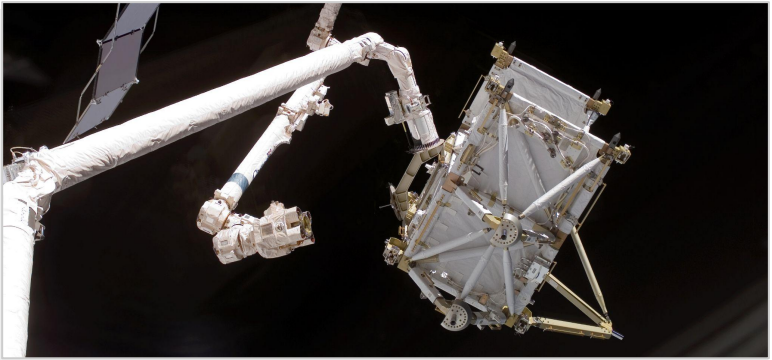


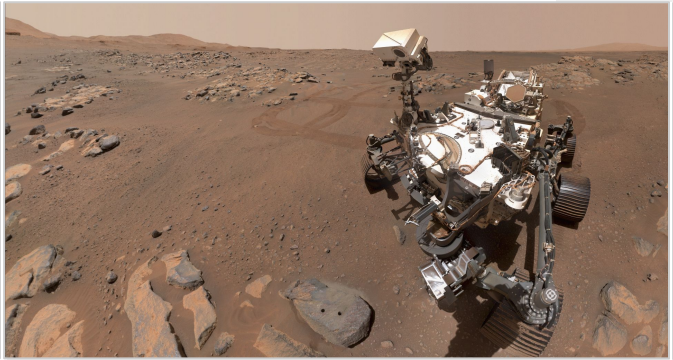


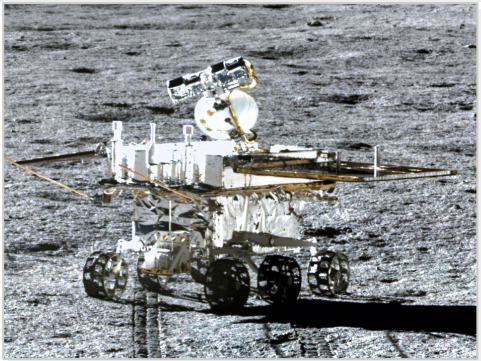




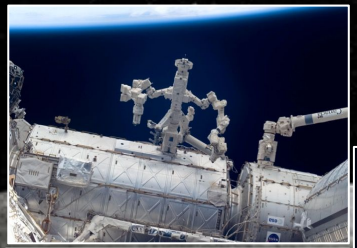
Robots are also in space





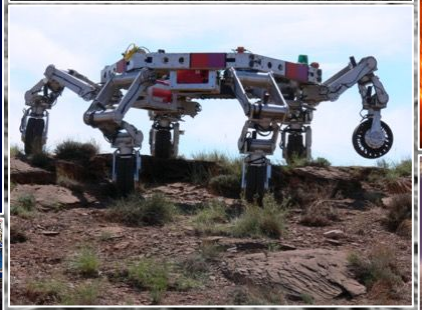


Manipulators



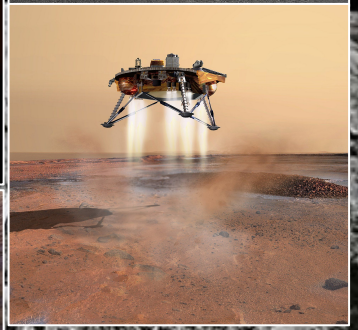
Humanoid robots

Mobility systems



Robotic spacecraft

Robotic landers



1970 – joystick

2020 – 2M lines of code

Increasing amount of software **+** cost of software development
= *demand for reuse*

The Demand for Reuse

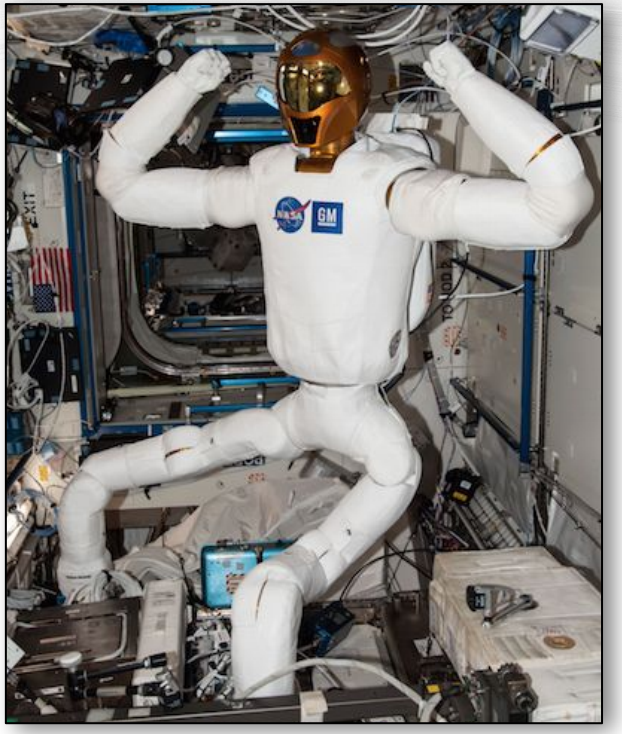
The space community is already moving toward componentized, reusable, and open frameworks for flight software and mission control

- F' (F Prime)
- core Flight System (cFS)
- Yamcs
- OpenMCT

Also using smaller open source projects in flight

- AprilTag (visual fiducial system) used on Perseverance

ROS-based robots have already been to space



2014: **Robonaut 2**

2019: **Astrobee**

NASA VIPER

Prospecting for lunar resources in permanently shadowed regions of the lunar south pole

- **ROS** used in ground software systems
- **Gazebo** simulation used in mission development, testing, planning, operator training, etc.
- Other open source software
  - cFS/ROS bridge
  - Yamcs
  - OpenMCT
- NASA requires software used in **flight missions** to be space qualified

spaceROS

What we need: A version of ROS for space applications!

A space-certifiable and reusable space robotics framework

- Support certification according to flight software standards, like DO-178C and NASA's NPR7150.2
- Provide artifacts to allow space flight projects to gain a head start on their certification efforts

That brings the benefits of ROS to space robotics

- Enable rapid development of new robotic capabilities
- Facilitate reuse across missions, reducing development effort and costs
- Open source software, use open community processes

What is Space ROS?

Space ROS 2022

Foundation

- Builds
- Releases
- Continuous Integration
- Maintenance
- Package subset
- Docker images
- Embedded target(s)

Tools and Processes

- Requirements tools and processes for traceability and analysis
- Code analysis tools with SARIF output
- Dashboard for issue navigation, visualization & dispositioning
- Development workflow
- Quality level(s)
- MC/DC testing

Space-Specific Functionality

- Eventing & Telemetry Subsystem
- C++ PMR allocator
- Sample applications for navigation and manipulation
- Simulation assets

Requirements management in **aerospace**

More than checklists

- Typically managed using a strict process and proprietary tools
  - Process is often according to some accepted standard, e.g. DO-178C
- Requirements must be complete - no software without requirements - and highly detailed
- Multiple levels of requirements - from abstract needs to detailed behaviour timings
- Traceability is essential - source to requirement to implementation and verification, and back again
- Requirements ultimately are used to support a certification process

Requirements management in **open source software**

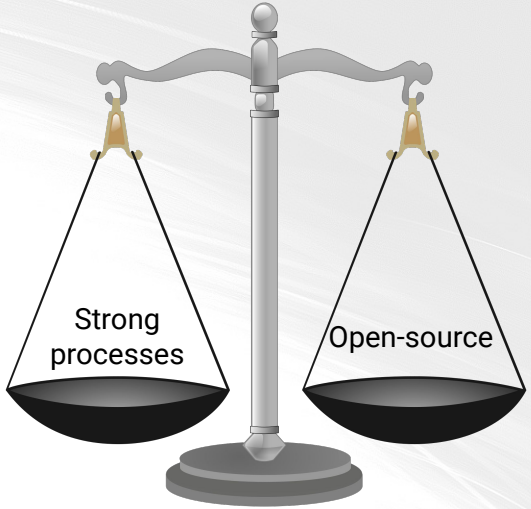
What requirements?

- Requirements are typically non-existent
- Any requirements that do exist are lightly managed (and easily get out-of-date)
- Heavy processes are shunned to avoid discouraging contributions

Open requirements for Space ROS

Balance competing forces

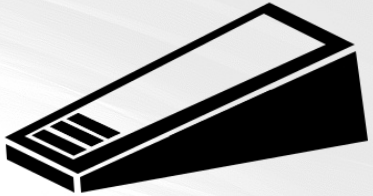
- Heavy-weight requirements process using expensive tools is inappropriate for an open-source project
- Need a process and tool(s) that won't discourage contributions
  - Contributors are unlikely to purchase expensive requirements management tools
  - Heavy-weight processes discourage drive-by contributors
- Must strike a balance between aerospace's need for strong processes and open-source's desire for ease-of-contributing



Tools for open requirements management

Doorstop

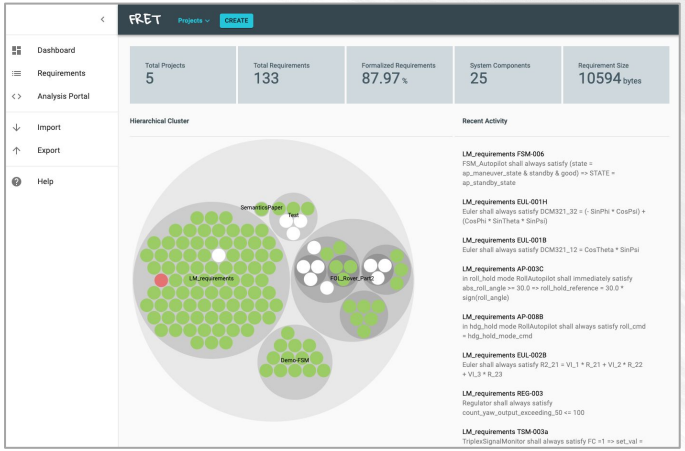
- Simple requirements management tool providing a command-line-and-text-editor based workflow
  - Add and edit requirements
  - Trace between requirements
  - Generate reports
- Based on YAML files stored in a versioned repository
  - Requirements are stored in a human-readable format
  - Easy to parse for additional automation tools
  - Requirements can be written in restricted natural language, e.g. EARS
- Open-source
  - Can be modified to meet our needs
  - Freely available to contributors
  - https://doorstop.readthedocs.io/en/latest/



Tools for open requirements management

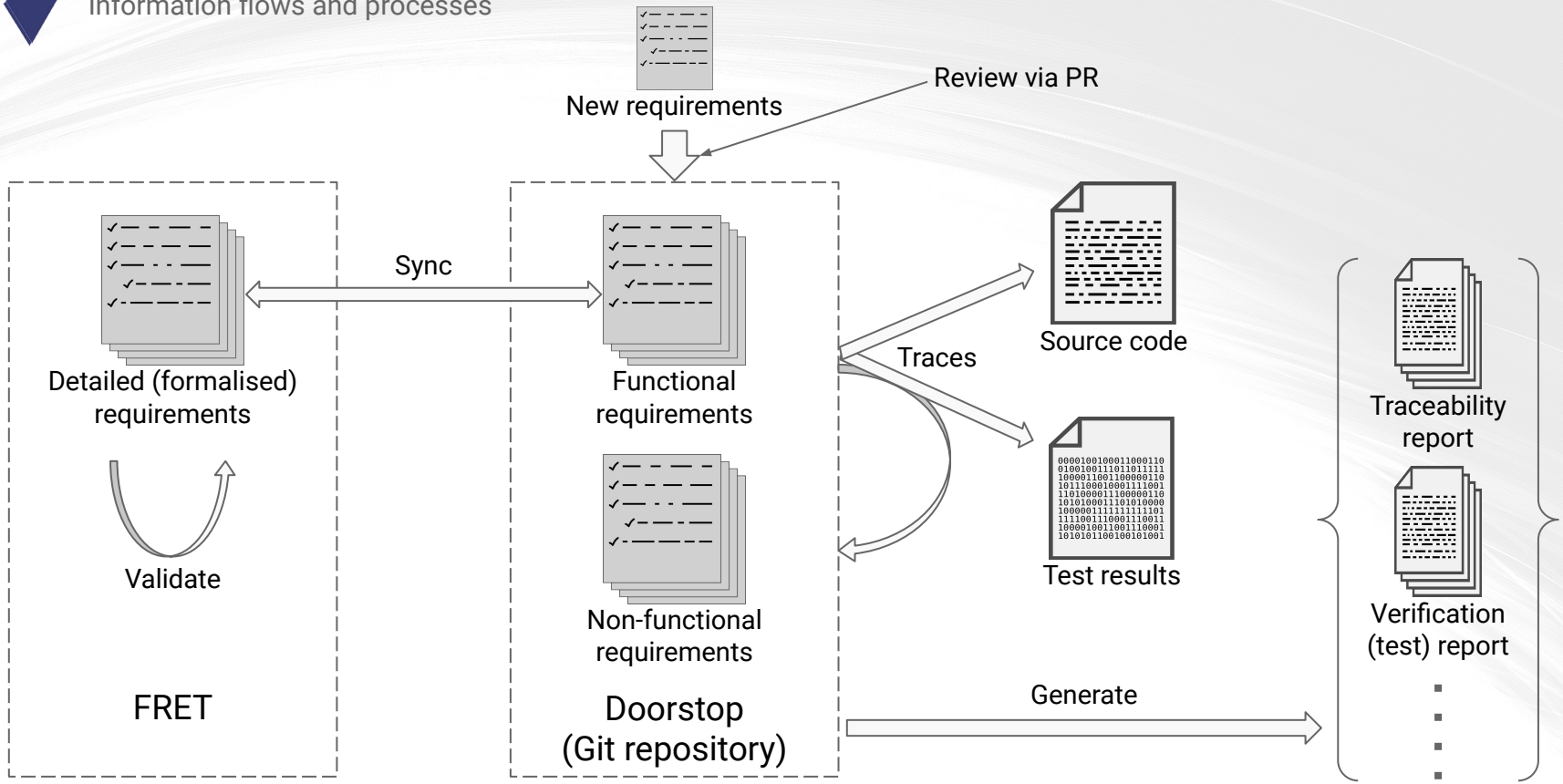
FRET

- Graphical tool for creating and managing semi-formal and formal requirements
- Stores requirements in a database, with JSON import/export
- Requirements can be written in “FRETish”, which can contain linear temporal logic expressions
- Automatic model checking of requirements for consistency and conflicts
- Although freely available, the learning curve is steeper than Doorstop
- Automatic generation of safety monitor(s) from requirements expressed FRETish



Management of requirements in Space ROS

Information flows and processes



Management of requirements in Space ROS

Key points

- Doorstop used for:
  - High-level requirements
  - Non-functional requirements
  - Requirements traceability management
  - Artefact generation (e.g. traceability reports)
- FRET used for detailed functional requirements and consistency checks
- Requirements stored in Git (single source of truth)
  - Pull requests provide a chance for requirements review
- Trace to implementation and tests via Git commit hashes

Static Analysis

Meeting the needs of aerospace with open-source analysers

- Increase code quality, provide information supporting verification efforts
- Space ROS provides a suite of static analyzers, including IKOS and Cobra from NASA
- Currently adding dynamic analysis: code coverage and MC/DC testing
- The static analysis tools generate SARIF output
  - Currently, most tools parse the output of the tool
  - Tools should eventually support SARIF directly; would allow for more detailed information in SARIF
- Filtering pass to remove (some) redundancy
  - Currently, removing identical issues
  - Would like to remove semantic equivalents
- The results are made available to the Space ROS Dashboard
  - An archive format that contains analyzer output, filtered output, and metadata

IKOS (Inference Kernel for Open Static Analyzers)

Application of formal methods to support certification

- DO-178C includes an extension, DO-333, that describes how developers can use static analysis in certification
  - DO-333 provides guidance on how formal (mathematical) methods may to produce verification evidence suitable for use in certification
  - DO-333 lists Abstract Interpretation as a suitable methodology
- IKOS is a static analysis framework, based on the Theory of Abstract Interpretation
  - Used to develop static analyses that are both precise and scalable
  - The framework makes it accessible to a larger class of static analysis developers
- References
  - https://jorgenavas.github.io/papers/ikos-sefm14.pdf
  - https://github.com/NASA-SW-VnV/ikos

"... computations can be abstracted and reduced to a generalized set of objects and still exhibit the same critical properties of the parent program. By reducing the set of objects through abstraction, IKOS is scalable to large complex computer programs and presents a sound approach to verification of such programs."

Cobra (code browser and analyzer)

An extensible, interactive tool for the analysis of C/C++ code

- A static analysis capability that works well for large code bases
- Fast analysis of general code patterns, common coding flaws, or coding rule compliance
  - Performs lexical analysis to generate a stream of language-level tokens
  - Stores the key information of source code in an extremely simple data structure
- Can be used in one of three modes
  - As an **interactive query engine** to match patterns with a simple query language
  - Execute **inline Cobra programs** that can contain arbitrary branching and iteration over the token stream to identify more complex types of patterns
  - As an infrastructure for building more elaborate **standalone checkers** that are compiled separately and linked with the Cobra code that builds the central data structure
- References
  - https://software.nasa.gov/software/NPO-50050-1
  - https://github.com/nimble-code/Cobra

Cobra (code browser and analyzer)

An extensible, interactive tool for the analysis of C/C++ code

```
spaceros-user@ba0b59ced39b:~$ ament_cobra --help
usage: ament_cobra [-h] [--include_dirs [INCLUDE_DIRS [INCLUDE_DIRS ...]]] [--exclude [EXCLUDE [EXCLUDE ...]]] [--ruleset RULESET] [--compile_cmds COMPILE_CMDS]
                   [--xunit-file XUNIT_FILE] [--sarif-file SARIF_FILE] [--cobra-version] [--verbose]
                   [paths [paths ...]]

Analyze source code using the cobra static analyzer.

positional arguments:
  paths                 Files and/or directories to be checked. Directories are searched recursively for files ending in one of '.c', '.cc', '.cpp', '.cxx'.
                        (default: ['.'])

optional arguments:
  -h, --help            show this help message and exit
  --include_dirs [INCLUDE_DIRS [INCLUDE_DIRS ...]]
                        Include directories for C/C++ files being checked.Each directory is passed to cobra as '-I<include_dir>' (default: None)
  --exclude [EXCLUDE [EXCLUDE ...]]
                        Exclude C/C++ files from being checked. (default: [])
  --ruleset RULESET     The cobra rule set to use to analyze the code: basic, cwe, p10, jpl, misra2012, C++/autosar. (default: basic)
  --compile_cmds COMPILE_CMDS
                        The compile_commands.json file from which to gather preprocessor directives. This option will take precedence over the --include_dirs
                        options and any directories specified using --include_dirs will be ignored. Instead, ament_cobra will gather all preprocessor options
                        from the compile_commands.json file. (default: None)
  --xunit-file XUNIT_FILE
                        Generate a xunit compliant XML file (default: None)
  --sarif-file SARIF_FILE
                        Generate a SARIF file (default: None)
  --cobra-version       Get the cobra version, print it, and then exit (default: False)
  --verbose             Display verbose output (default: False)
spaceros-user@ba0b59ced39b:~$
```


SARIF (Static Analysis Results Interchange Format)

Unification of static analysis results

- A JSON-based exchange format for the output of static analysis tool
- Used by IDEs, code analysis tools, continuous integration systems, etc.
- SARIF output by all Space ROS static analyzers

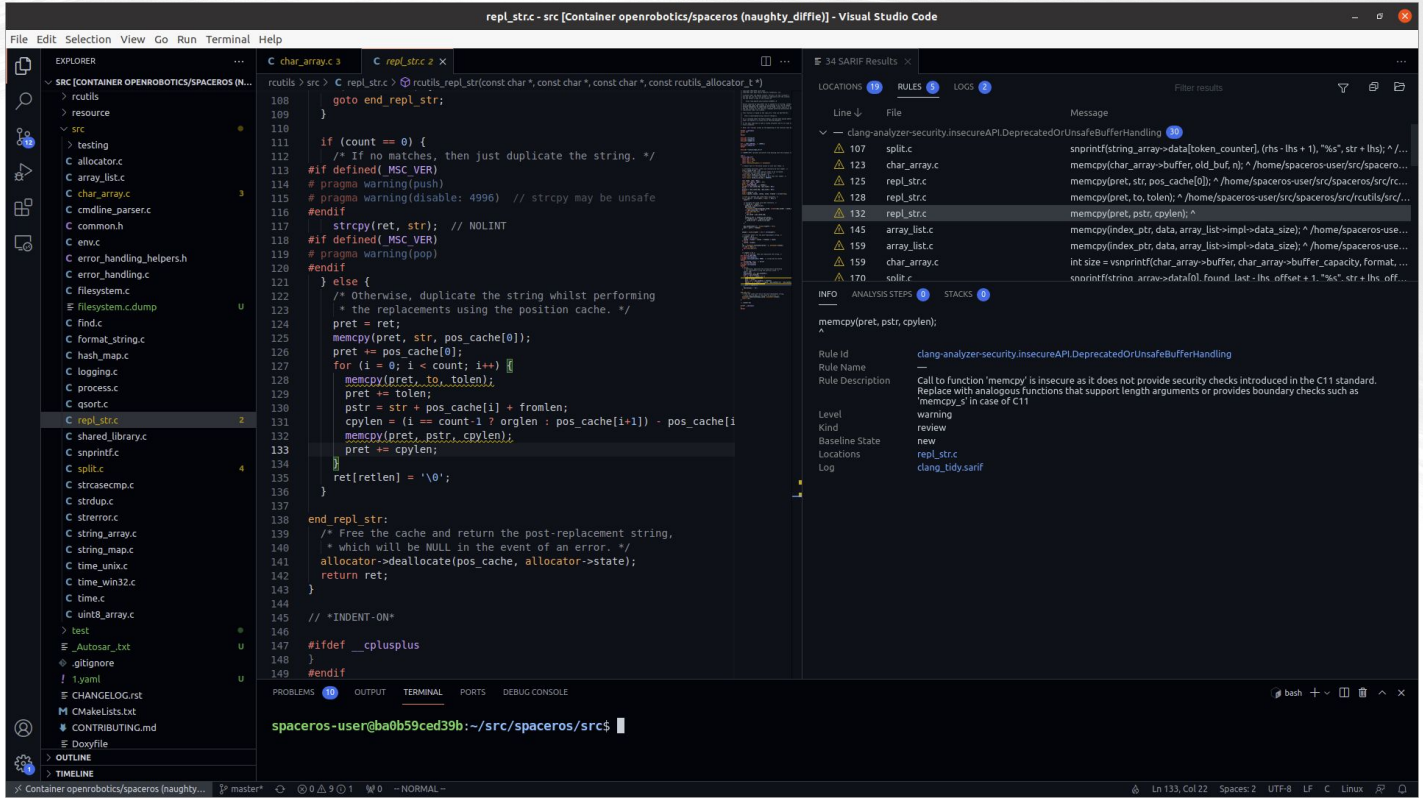
```
{
  "version": "2.1.0",
  "$schema": "http://json.schemastore.org/sarif-2.1.0-rtm.5",
  "properties": {
    "comment": "clang-tidy output converted to SARIF by ament_clang_tidy"
  },
  "runs": [
    {
      "tool": {
        "driver": {
          "name": "clang-tidy",
          "version": "10.0.0",
          "informationUri": "https://clang.llvm.org/extra/clang-tidy/",
          "rules": []
        }
      },
      "artifacts": [],
      "results": []
    }
  ]
}

```

https://docs.oasis-open.org/sarif/sarif/v2.0/sarif-v2.0.html

VSCode SARIF plugin

Making static analysis results visible



Extending the VSCode SARIF plugin

Making static analysis results visible

- Insight into static analysis, code coverage, build status, issue burndown, etc.
- A starting point for the open source community to extend and improve
- Interface to build, test, using Earthly (same as CI)
- Integrate with external dispositioning systems
- Plugin available on the VSCode Marketplace

Ongoing development

Space ROS 2023+

Foundation

- Regular releases

Tools and Processes

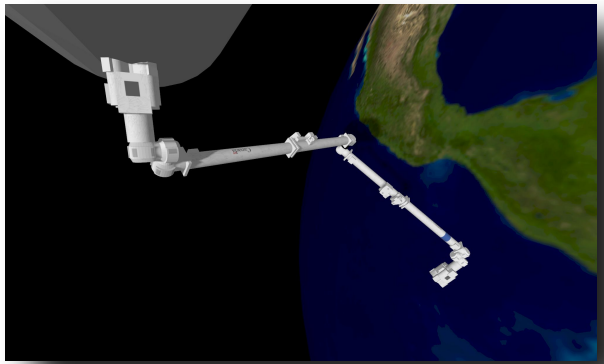
- Dashboard (continued)
- Auditing support, checklists, reports
- Code improvements
- Back-porting requirements
- Requirements analysis

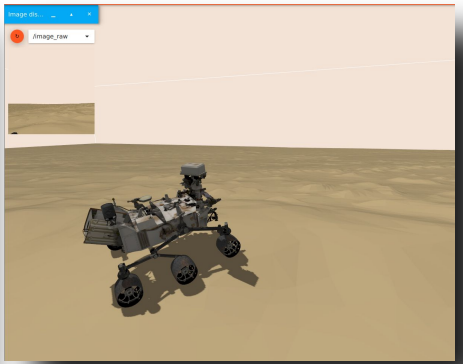
Space-Specific Functionality

- cFS/ROS 2 bridge
- Applications

Open processes and artefacts for community-driven validation

- We're integrating open source tools and processes to help improve software quality
  - Requirements, code analysis, developer workflow, quality levels
- This is done in the context of Space ROS, but could be useful to other domains
- We welcome your contributions and input
- https://github.com/space-ros






spaceROS

Please fill out the ROS and Gazebo User Survey!

Humble						
EOL ROS2 Distros						

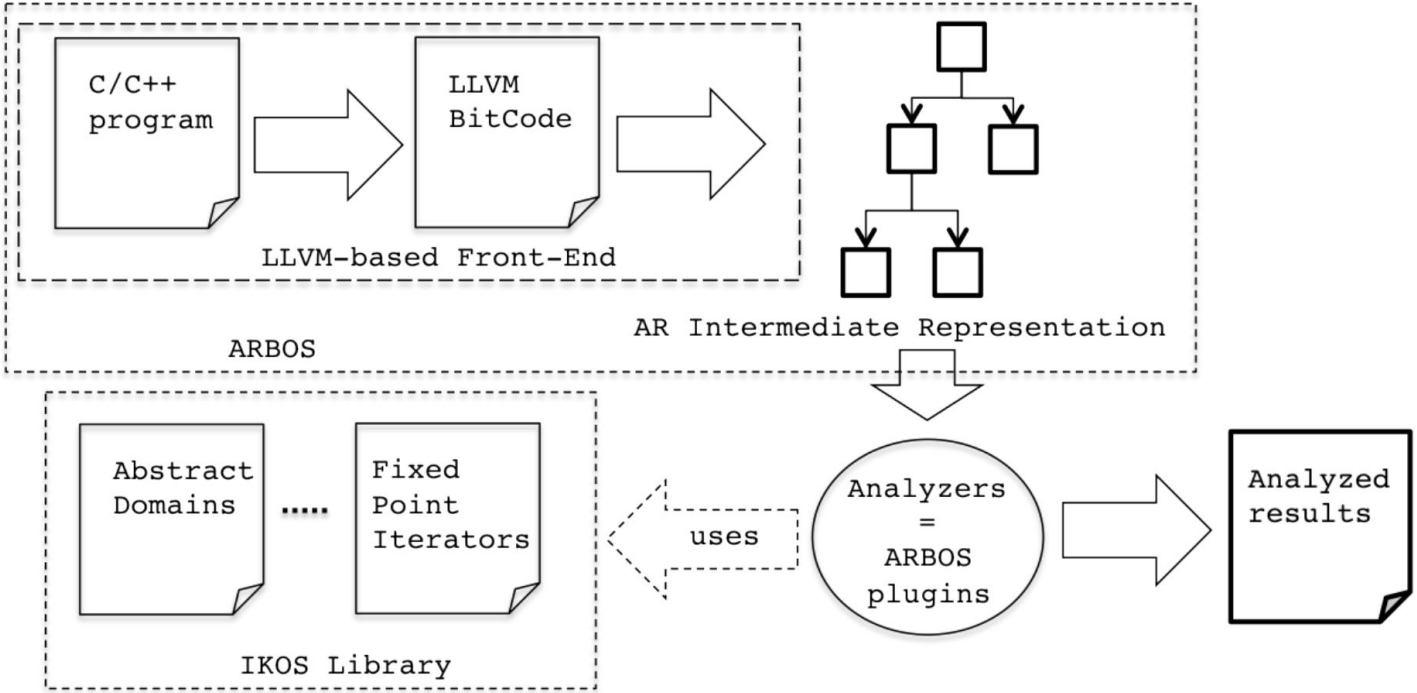
SCAN ME

Select all Gazebo distril … ure of that experience.

	Created packages			Developing a product	Other
Citadel					
Fortress					
Garden					

IKOS (Inference Kernel for Open Static Analyzers)

The IKOS framework architecture



https://jorgenavas.github.io/papers/ikos-sefm14.pdf