Find, Use, and Conserve Tools for Formal Methods

git: https://gitlab.com/sosy-lab/benchmarking/fm-tools

web: https://fm-tools.sosy-lab.org

Dirk Beyer
LMU Munich, Germany

June 25, 2024, at Podelski Fest @PLDI 2024

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

SoSy-Lab Software Systems

Congratulations to Andreas Podelski

- Congratulations to Andreas to your birthday and to your great research achievements!!
- Not only theoretical results, but also formal-methods tools.
- Huge impact!
- Examples: Ultimate family of tools for software verification SMT solver SMTinterpol
- Excellent features and performance, medals in competitions

Open problem: It could be that one day,
Andreas does not come to university anymore.
What happens to those brilliant tools?

Vision

- All tools for formal methods work together to solve hard verification problems and make our world safer and more secure.
- Model checkers and theorem provers can be integrated into the software-development process as seamless as unit testing today.
- Model checkers, theorem provers, SMT solvers, and testers use common interfaces for interaction and composition.

Some Steps Towards the Vision

- **Find**: Which tools for software verification exist?
- ... for test-case generation?
- ... for SMT solving?
- ... for hardware verification?
- **Reuse**: How to get executables?
- Where to find documentation?
- Am I allowed to use it?
- How to use them?
- **Conserve**: Which operating system, libraries, environment?

Requirements for Solution

- Support documentation and reuse
- Easy to query and generate knowledge base
- Long-term availability/executability of tools
- Must come with tool support
- Approach must be compatible with competitions

Solution

One central repository:
https://gitlab.com/sosy-lab/benchmarking/fm-tools
which gives information about:

- Location of the tool (via DOI, just like other literature)
- License
- Contact (via ORCID)
- Project web site
- Options
- Requirements (certain Docker container / VM)
- Limits

Maintained by annual competition participants

Example: Entry for UAutomizer

```yaml
name: UAutomizer
input_languages:
  - C
project_url: https://ultimate-pa.org
repository_url:
    https://github.com/ultimate-pa/ultimate
spdx_license_identifier: LGPL-3.0-or-later
benchexec_toolinfo_module: ultimateautomizer.py
fmtools_format_version: "2.0"
fmtools_entry_maintainers:
  - danieldietsch
```

Example: UAutomizer's Contacts

```yaml
maintainers:
  - orcid: 0000-0003-4252-3558
    name: Matthias Heizmann
    institution: University of Freiburg
    country: Germany
    url:
       https://swt.informatik.uni-freiburg.de/staff/heizmann
  - orcid: 0000-0003-4885-0728
    name: Dominik Klumpp
    institution: University of Freiburg
    country: Germany
    url:
       https://swt.informatik.uni-freiburg.de/staff/klumpp
  - orcid: 0000-0002-5656-306X
    name: "Frank␣Schuessele"
    institution: University of Freiburg
    country: Germany
    url:
       https://swt.informatik.uni-freiburg.de/staff/schuessele
  - orcid: 0000-0002-8947-5373
    name: Daniel Dietsch
    institution: University of Freiburg
    country: Germany
    url:
       https://swt.informatik.uni-freiburg.de/staff/dietsch
```

Example: UAutomizer's Versions

```
versions:
  - version: svcomp24
    doi: 10.5281/zenodo.10203545
    benchexec_toolinfo_options: [--full-output]
    required_ubuntu_packages:
      - openjdk-11-jre-headless
    base_container_images:
      - docker.io/ubuntu:22.04
    full_container_images:
      - registry.gitlab.com/.../.../.../user:2024
```

Example: UAutomizer's Participation

```
competition_participations:
  - competition: SV-COMP 2024
    track: Verification
    tool_version: svcomp24
    jury_member:
      orcid: 0000-0003-4252-3558
      name: Matthias Heizmann
      institution: University of Freiburg
      country: Germany
      url: https://swt.informatik.uni-freiburg...
```

Example: UAutomizer's Documentation

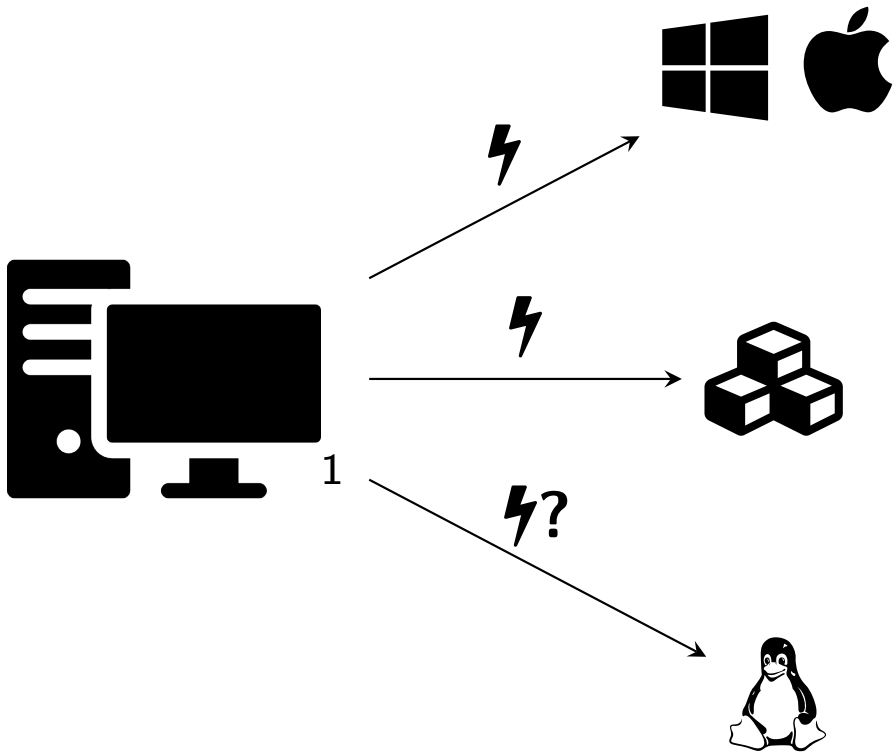
```
techniques:
  - CEGAR
  - Interpolation
  - Automata-Based Analysis
  - ...

literature:
  - doi: 10.1007/978-3-642-39799-8_2
    title: "Software␣Model␣Checking␣for␣People␣Who␣
       Love␣Automata"
    year: 2013
  - doi: 10.1007/978-3-031-30820-8_39
    title: "Ultimate␣Automizer␣2023␣(Competition␣
       Contribution)"
    year: 2023
```

FM-Tools is FAIR

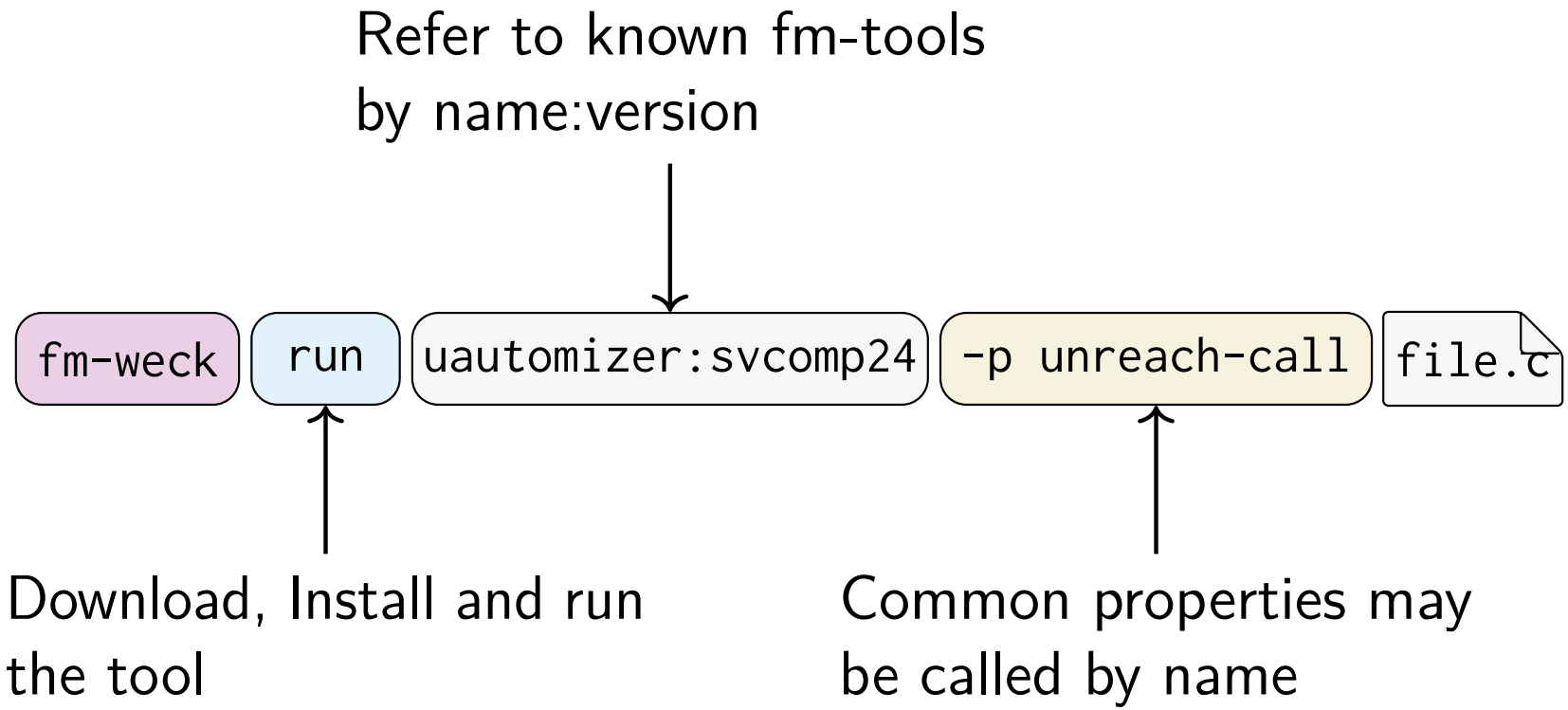
- **F**indable:
  overview is available on internet,
  generated knowledge base
- **A**ccessible:
  data retrievable via Git, format is YAML
- **I**nteroperable:
  Format is defined in schema,
  archives identified by DOIs, researchers by ORCIDs
- **R**eusable:
  Data are CC-BY, each tool comes with a license,
  format of tool archive standardized

What about the Environment?

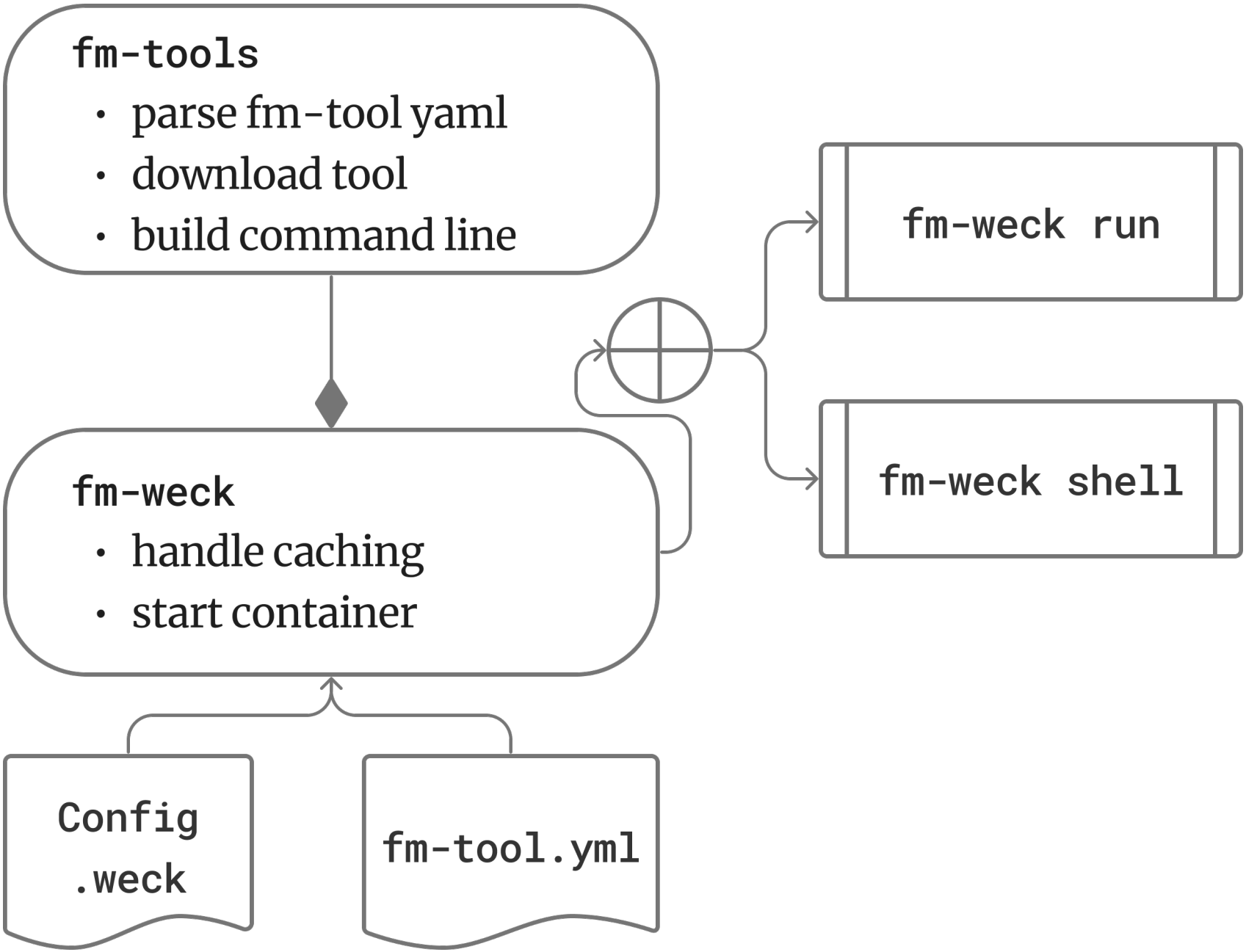


[1] Image: Flaticon.com

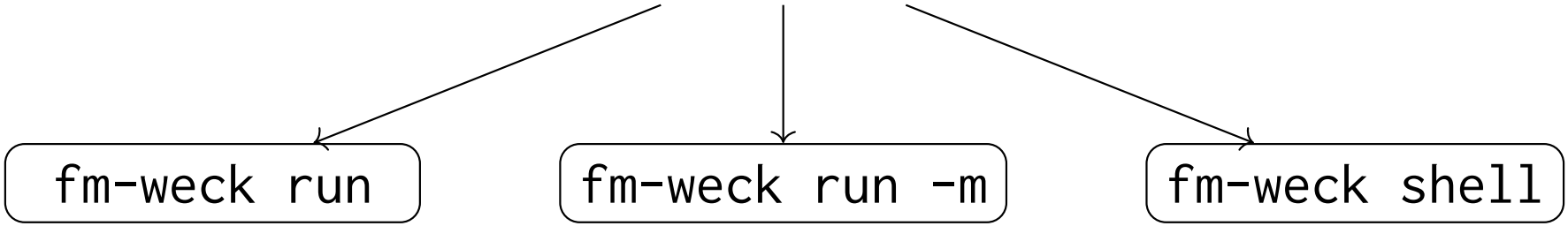
FM-Weck: Run Tools in Conserved Environment



- No knowledge of the tools CLI needed
- Tool runs in a container (no dependencies on host system)


fm-tools
• parse fm-tool yaml
• download tool
• build command line
fm-weck
• handle caching
• start container
Config
.weck
fm-tool.yml
fm-weck run
fm-weck shell

fm-weck

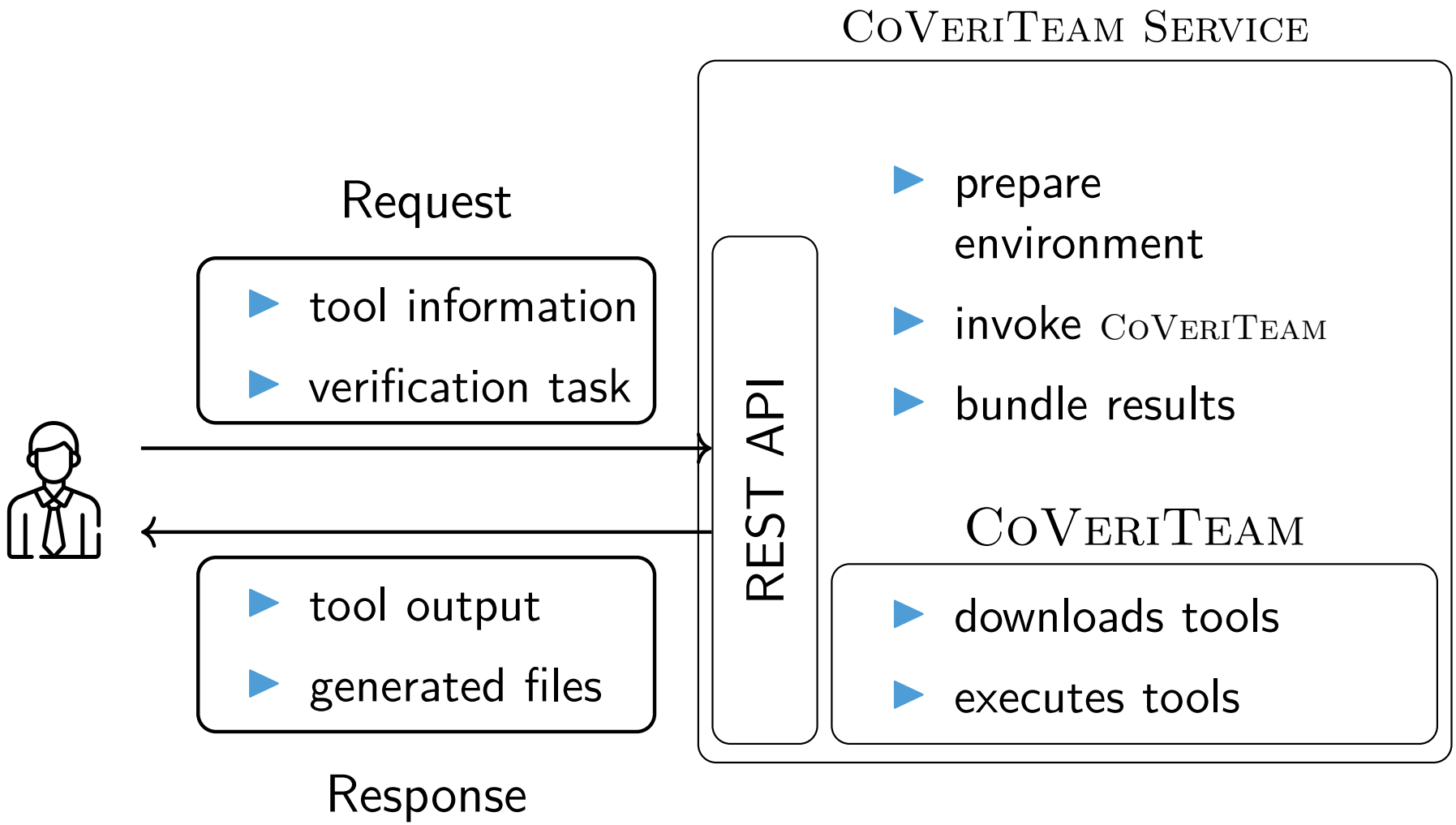


- Download and execute tool in container
- No knowledge of tool needed

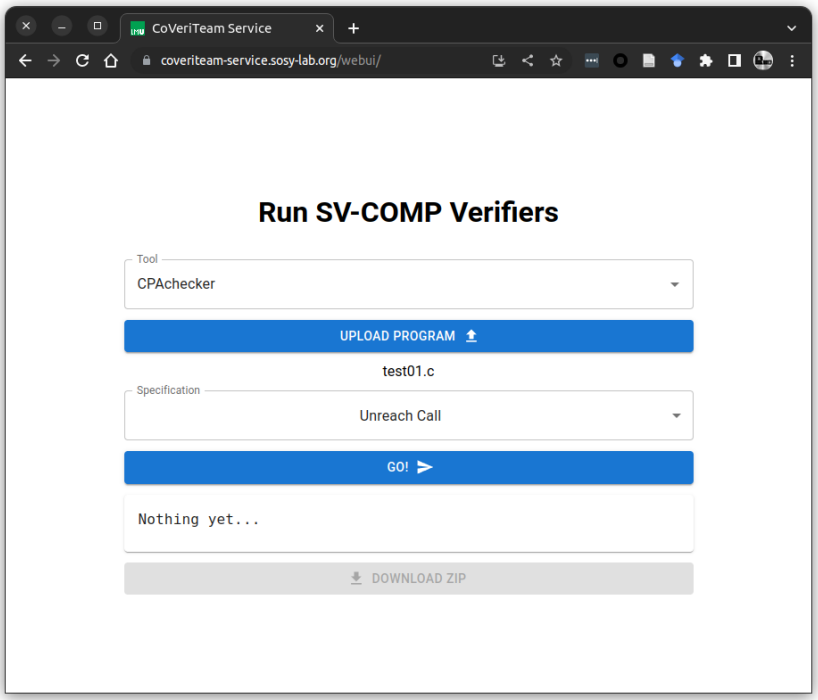
- Download and execute tool in container
- Expert knowledge about tool required

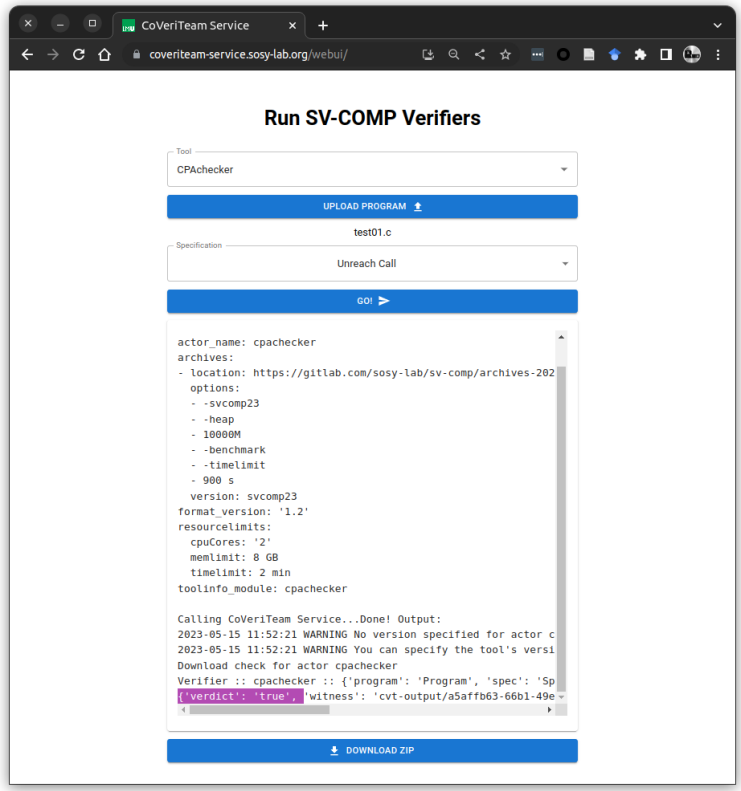
- Spin up interactive shell in tool environment

CoVeriTeam Service: Run Tool as Web Service



CoVeriTeam Service: Run Tool in Web UI





Conclusion

FM-Tools collects and stores essential information to:

- Run a tool as web service via CoVeriTeam Service
- Run a tool in conserved environment via FM-Weck
- Generate a knowledge base about formal-methods tools
  https://fm-tools.sosy-lab.org
- Organize competition participations

https://gitlab.com/sosy-lab/benchmarking/fm-tools