Effective Static Analysis Shift Left for the Development of Safe and Secure Embedded Systems

Roberto Bagnara, Abramo Bagnara

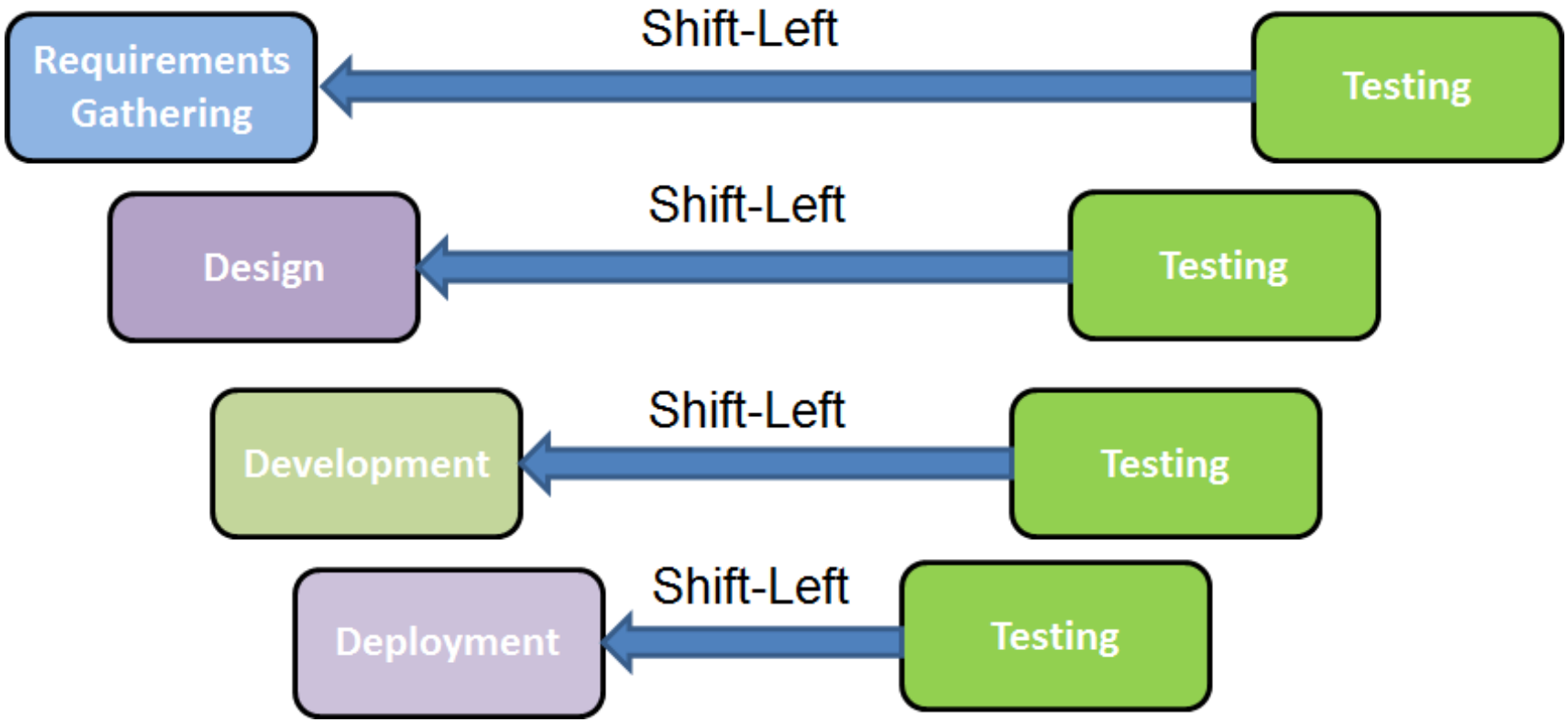
bugSeng
http://bugseng.com

& Applied Formal Methods Laboratory
Department of Mathematical, Physical and Computer Sciences
University of Parma, Italy

IWES 2022
Bari, September 23rd, 2022

Outline I

Shift-left


Requirements Gathering
Shift-Left
Testing
Design
Shift-Left
Testing
Development
Shift-Left
Testing
Deployment
Shift-Left
Testing

Shift-left: Not a New Idea

Shift-left dates back to 2001:

> *"Shift-left testing is how I refer to a better way of integrating the quality assurance (QA) and development parts of a software project. By linking these two functions at lower levels of management, you can expand your testing program while reducing manpower and equipment needs — sometimes by as much as an order of magnitude."*
>
> — Larry Smith, Dr. Dobb's Journal, September 1st, 2001

Shift-left: Not Just for Classical Testing

Today the notion of *testing* has to be interpreted in a broad sense, including all sorts of static and dynamic analysis techniques

$$\text{SAST} \Longrightarrow \begin{cases} \text{Static Application Security Testing} \\ \text{Static Application Safety Testing} \end{cases}$$

Software issues are better discovered earlier rather than later:

> *"Bugs are cheap when caught young."*
>
> — Larry Smith, ibidem

Example: The MISRA C/C++ Coding Standards

The MISRA coding standards are the most authoritative sets of guidelines for the development of safe and secure systems in C/C++

The highest payoff from the adoption of the MISRA Guidelines is achieved when they are adopted at the very beginning of a project...

...and it is systematically enforced with the help of a high-quality tool

Imposing MISRA C on an existing code base with a proven track record may be counterproductive if not done properly...

...this requires significant expertise and tools of even higher quality (powerful deviation mechanisms, baselining, ...)

Example: The MISRA C/C++ Coding Standards (cont'd)

Early adoption of the MISRA Guidelines is not greatly facilitated thanks to the use of Continuous Integration (CI) systems:

- Jenkins
- GitLab CI
- Bamboo
- ...

Basic functionality: whenever a change is committed into the source code repository, static analysis is triggered, and developers have access to the analysis results via the web

Important Features Coming from the CI System

- Keeping track of the last k analyses
- Freezing the results of particular analyses
- Showing trends (e.g., in the number of diagnostic messages)
- Triggering a new analysis (if not automatically done at each commit)
- Quality gates on the submitted patches, e.g.. developers commit to a "triage area" and this triggers execution of the static analysis tool
  - if the number of violations for the selected set does not increase, then the change is automatically merged into the development branch
  - otherwise an email alert is sent to the author of the change

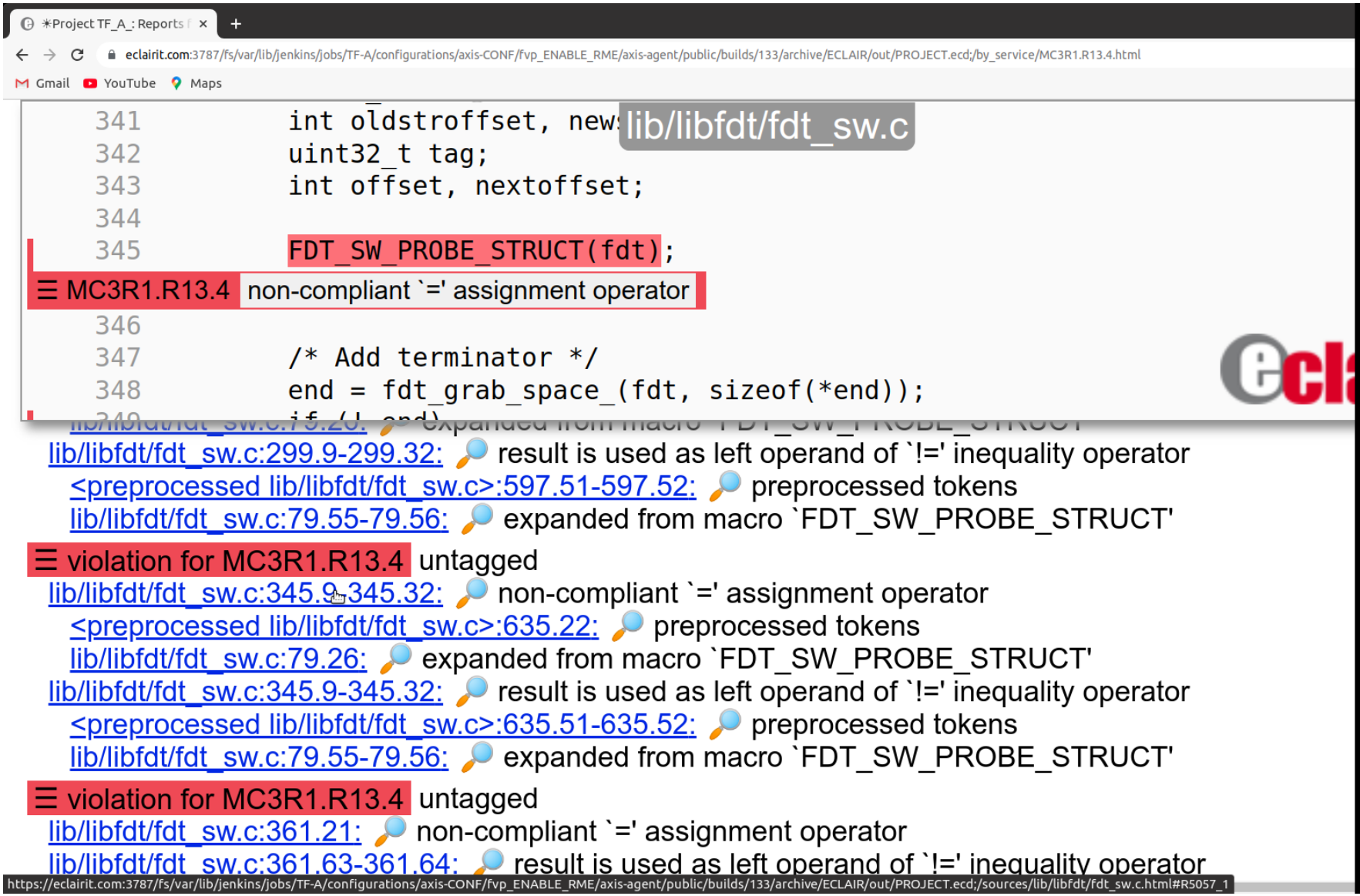
Key Features of the Static Analysis Software

Among the advantages of depoloying SAST via CI is the fact that users are spared from configuration and execution of the static analysis tool

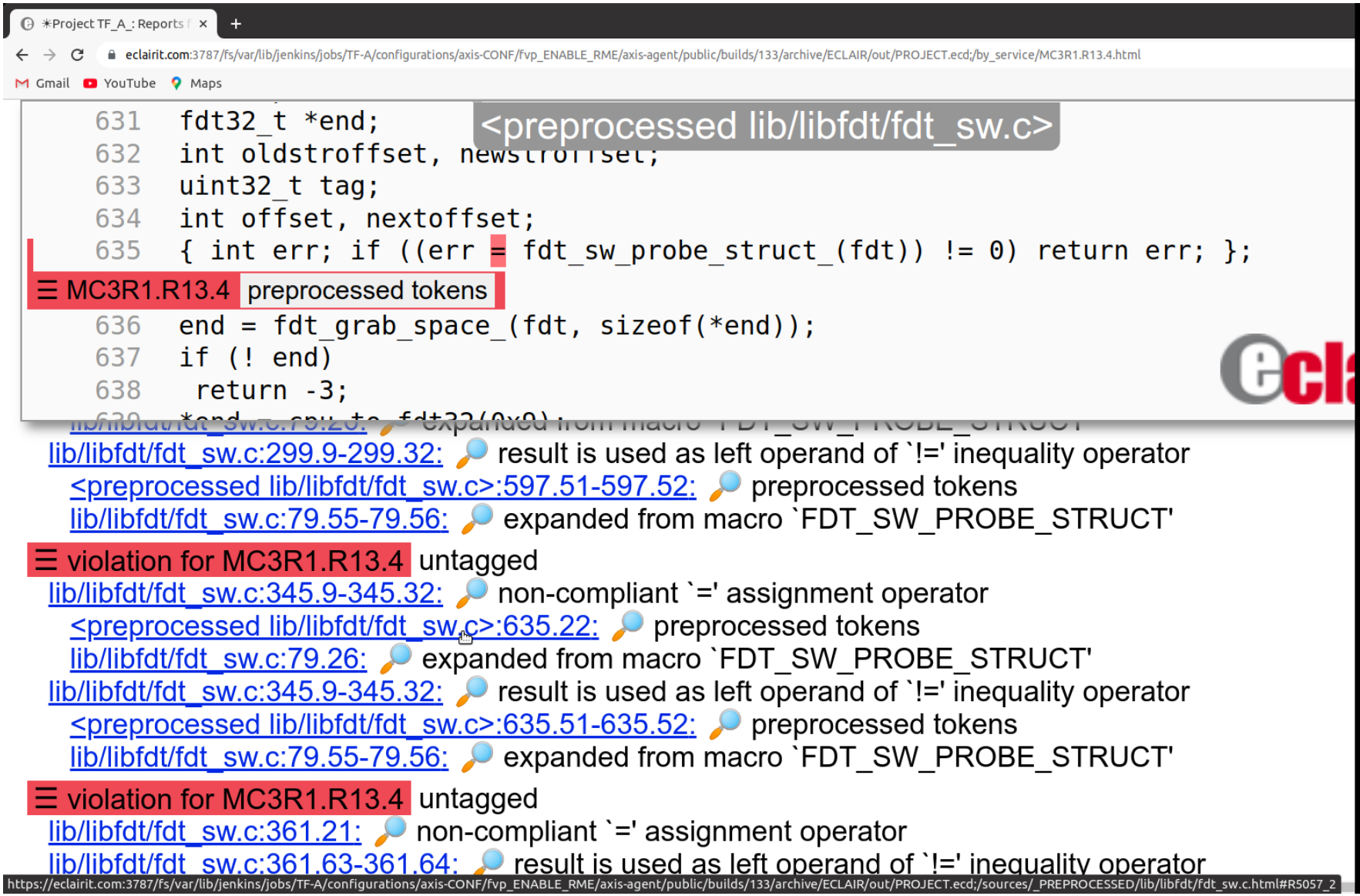
Other features are crucial for successful adoption, and only the static analysis tool can provide them (or not):

1. Users must have access to fully detailed reports (e.g., Sarif as implemented, e.g., in GitHub is inadequate)
2. Each user must be able to use private, sophisticated filters (i.e., locally-stored and independent from one another)
3. Each user must be able to use hisher favorite IDE (Eclipse, Visual Studio, Visual Studio Code, NetBeans, CLion, . . .)

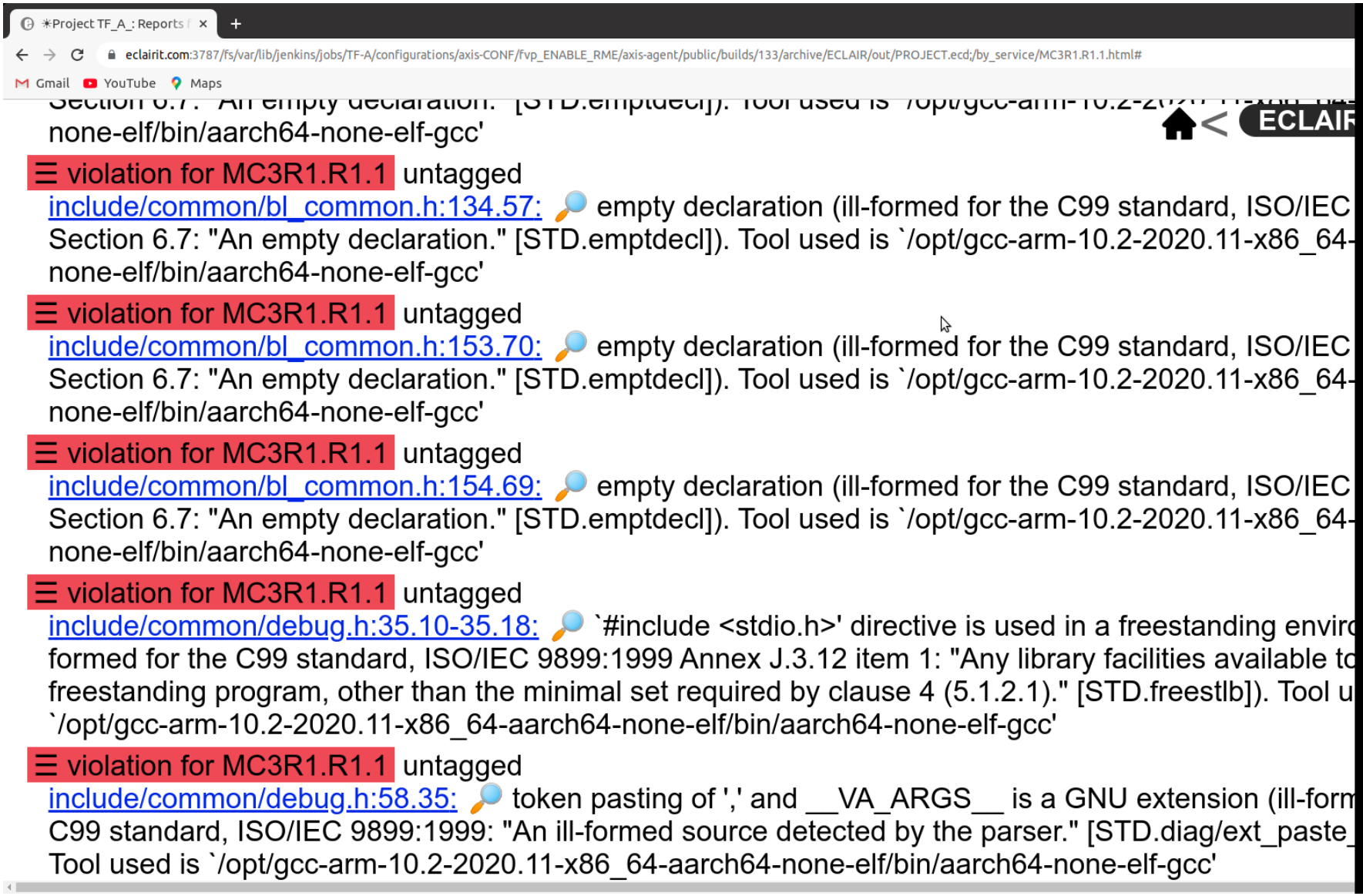
Access To Fully-Detailed Reports from Anywhere



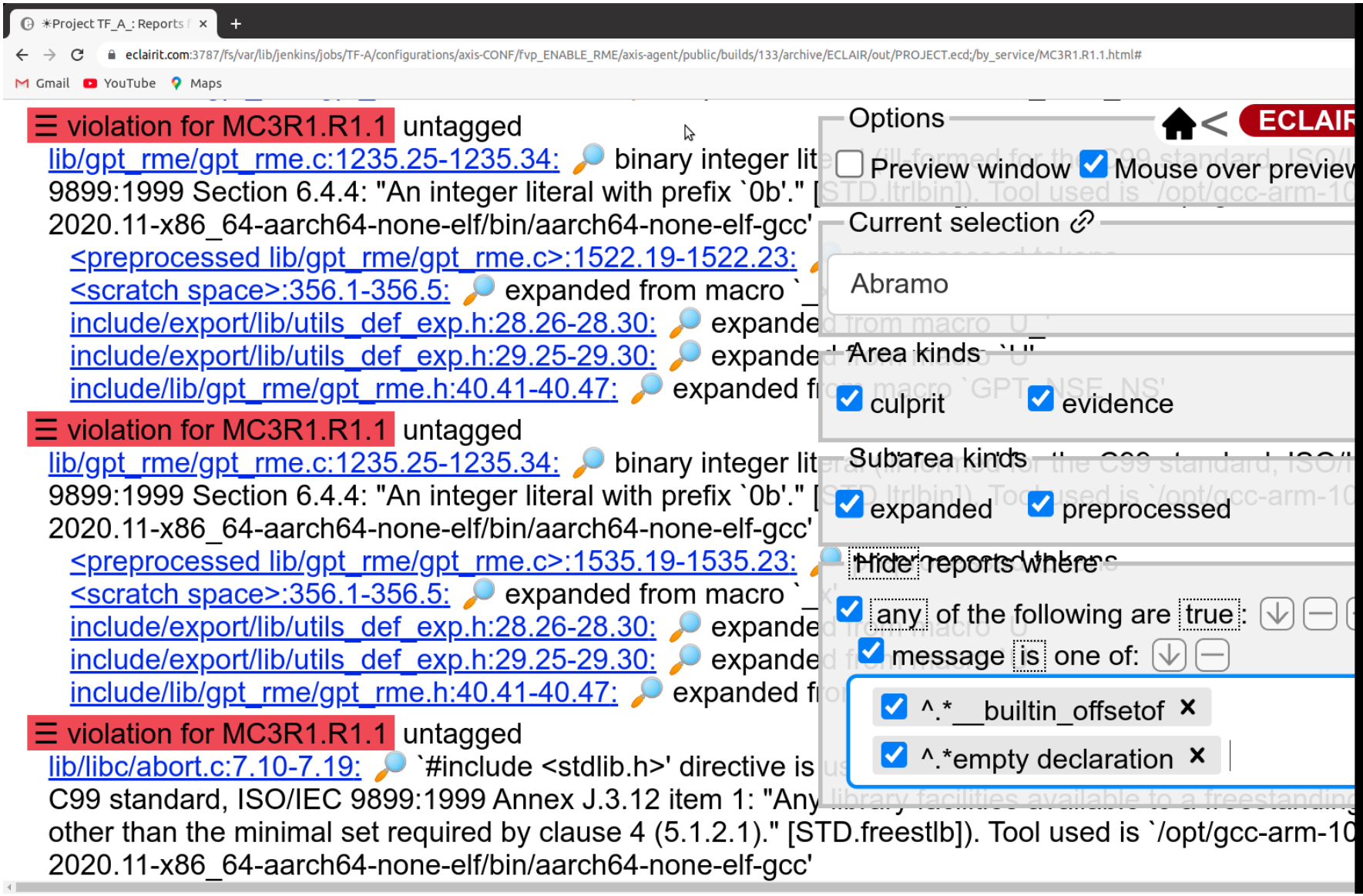
Access To Fully-Detailed Reports from Anywhere (cont'd)



Individual Filtering Capabilities



Individual Filtering Capabilities (cont'd)

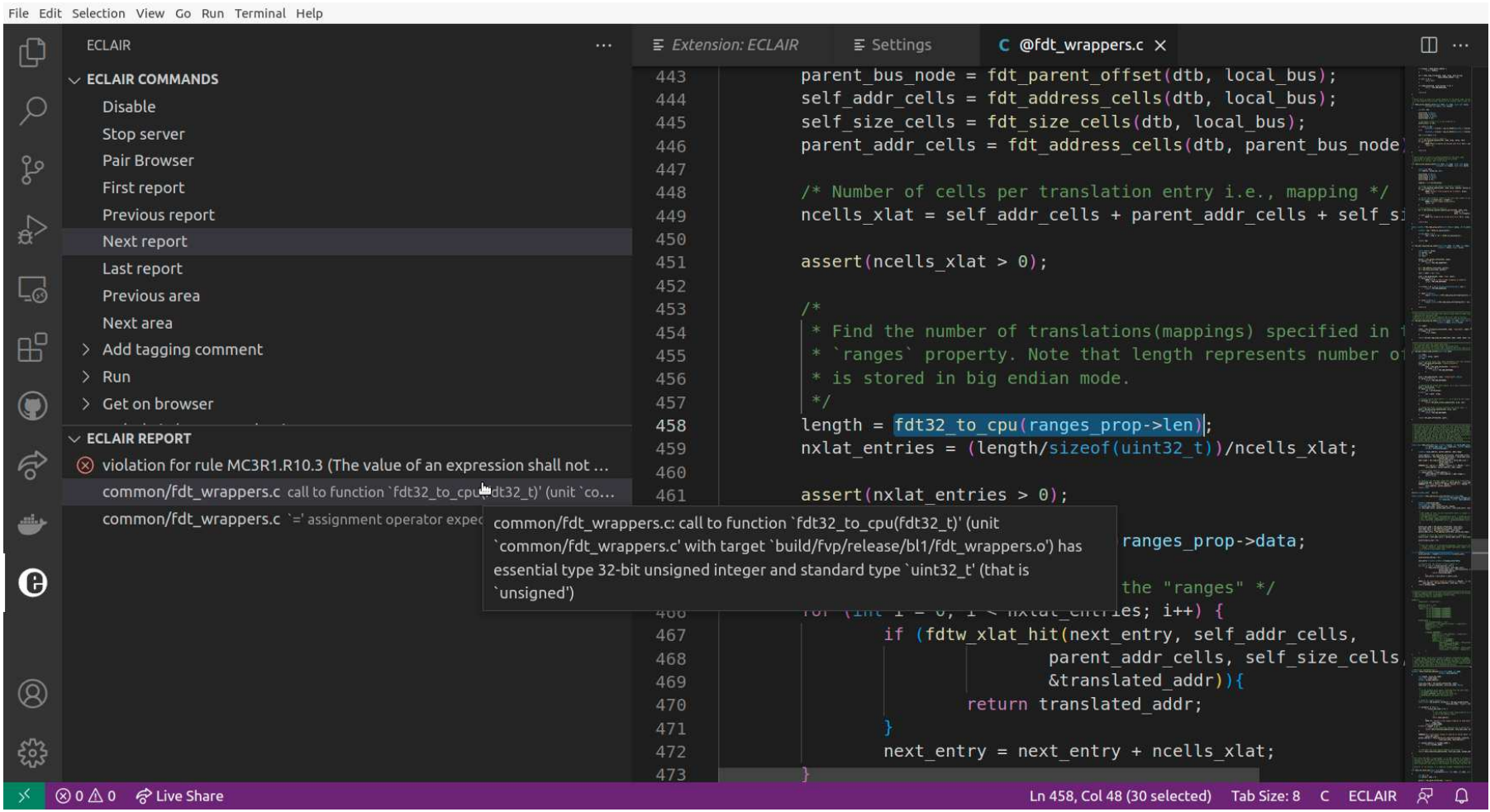


Use of Favorite IDEs

Users must be able to use their favorite IDE to act upon the analysis findings, which entails:

1. The IDE plugin must be able to relate the local source code and the remotely analyzed source code
2. The IDE plugin must take into account line insertions and removal to adapt the location information of the involved code areas

Use of Favorite IDEs (cont'd)



Conclusion

Shift-left adoption of static analysis tools is highly beneficial

- it reduces reworking
- it decreases time to completion

If the static analysis tool supports high-quality output delivery over the network, much can be done by the CI system itself

But for some key fetures the static analysis tool must have been designed carefully and with remote deployment in mind

ECLAIR has recently been chosen as the solution for static analysis in Continuous Integration systems of Trusted Firmware

Visit `eclairit.com` and make up your mind on the subject

Download a trial version of the ECLAIR Client Kit to play with the IDEs integrations

eclairit.com

eclairit
WORK IT BETTER MAKE IT STRONGER

On eclairit.com you can see what it looks like to use ECLAIR on an integration server and browse the reports directly in your browser, without having to install ECLAIR on your PC. No registration is required. Several popular open-source projects are here analyzed for compliance with MISRA and other coding standards. Among those: Linux-next, bitcoin, NASA Core Flight System, ARM Mbed OS, ARM Trusted Firmware A, XEN hypervisor, Zephyr RTOS.

JENKINS

SEE ECLAIR IN ACTION WITH JENKINS

GITLAB

SEE ECLAIR IN ACTION WITH GITLAB

ECLAIR CLIENT KIT

Want to use this service from within your favorite IDE?

TRY ECLAIR CLIENT KIT

The End


Questions?
roberto.bagnara@unipr.it
info@bugseng.com
bugseng