A «NEW» STATIC ANALYZER: THE COMPILER

Maurizio Martignano
Spazio IT – Soluzioni Informatiche s.a.s
Via Manzoni 40
46030 San Giorgio Bigarello, Mantova
https://www.spazioit.com

Agenda

Agenda

- Need for Speed
- Libraries, Libraries and again Libraries (libadalang)
- Clang/LLVM – SonarQube
- SAFe Toolset
- Future Activities

Need for Speed

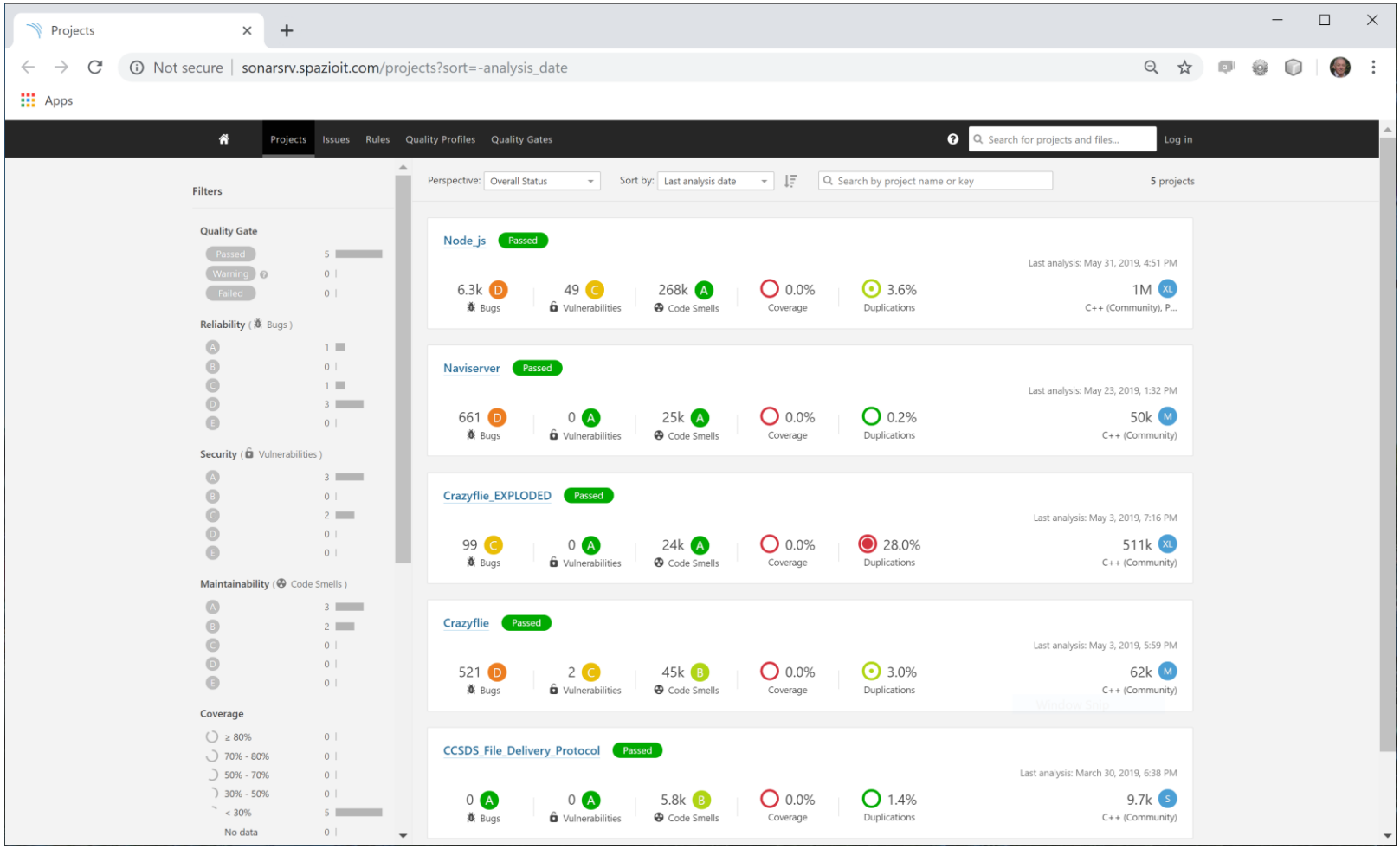
Need for Speed

- The size of software codebases is increasing dramatically:

Year	System	Size
1974	F16A Plane	135 K
1981	Space Shuttle PFS	400 K
2008	ESA ATV	1 M
2012	NASA Curiosity	2.5 M
2012	F35 Plane	10 M
Nowadays	Car	10-150 M

- Compilers and Static Analyzers need to be fast and efficient (i.e. able to “digest” large codebases in a reasonable time).

Need for Speed

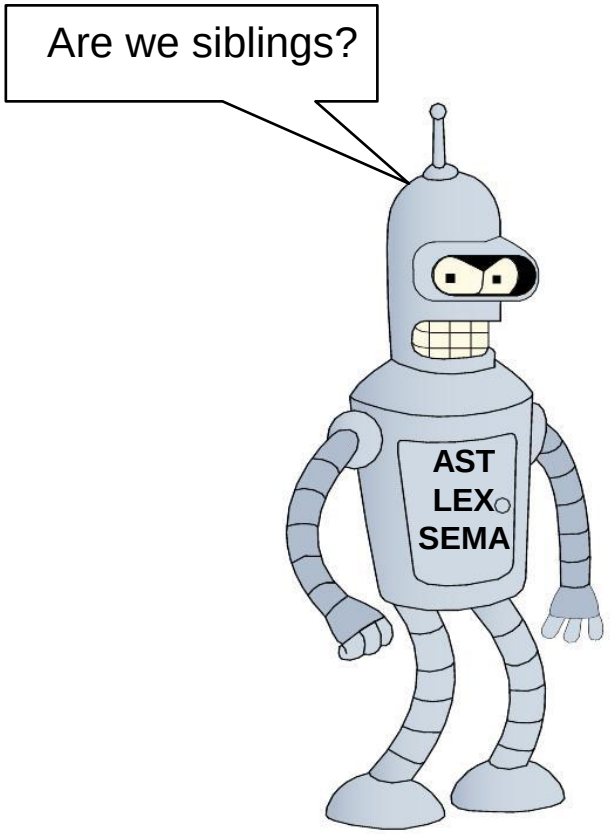


Need for Speed

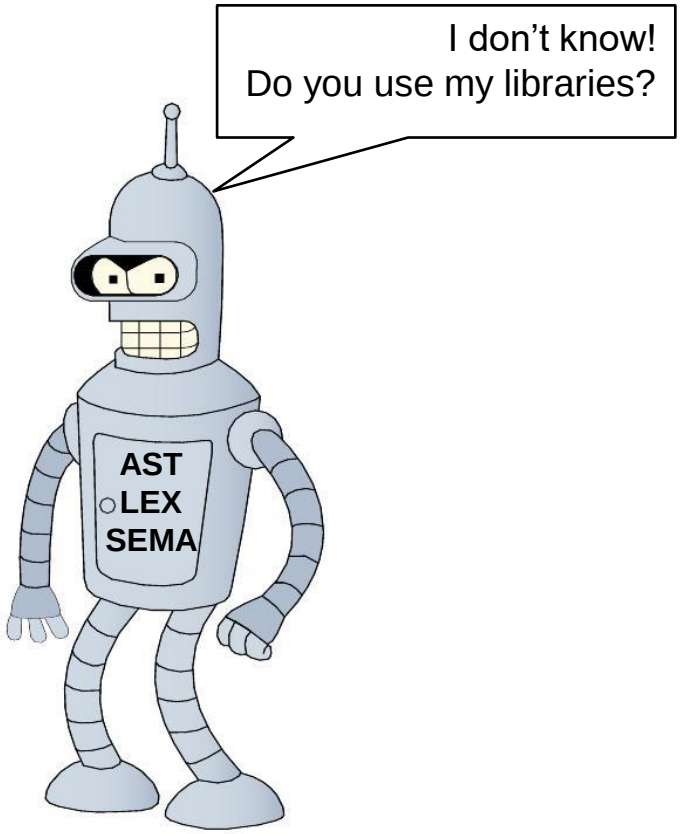
- Deep vs. Shallow Parsing

- Unforgiving vs. Forgiving Parsing

Libraries, Libraries and Again Libraries



Static Analyzer



Compiler

Libraries, Libraries and Again Libraries

- Suppose that for a given language we have a compiler and a static analyzer that are two separate software products, using different libraries and technologies (each one of them as its own lexer, parser, semantic analyzer and so on).
- Suppose the developer community behind that language and tools is not very big and doesn't have many resources, lots of energy.
- In case the language changes, evolves, for whatever reason, which of the two tools (the compiler or the static analyzer) will keep up with the language evolution?
- In the same way, which of the two tools will be more performant?

Libraries, Libraries and Again Libraries

- PC-Lint does not support the latest C/C++ Standards.
- Frama-C Semantic Analyzer cannot process all C/C++ constructs.
- Cppcheck sometimes stops when “digesting” “strange” codebases (e.g. Brotli).
- Ada ASIS does not support Ada 2012 (but the GNAT compiler does).
- In the Ada “libadalang” GitHub website we have: “Libadalang does not (at the moment) provide full legality checks for the Ada language. If you want such a functionality, you’ll need to use a full Ada compiler, such as GNAT.”
- and so on…

Libraries, Libraries and Again Libraries

- "The LLVM Project is a collection of modular and reusable compiler and toolchain technologies. (…) The LLVM Core libraries provide a modern source- and target-independent optimizer, along with code generation support for many CPUs. (…) Clang is an LLVM native C/C++/Objective-C compiler, which aims to deliver amazingly fast compiles."

- In fewer words Clang/LLVM is a compilation toolchain where absolutely everything is built in a modular fashion as collection of reusable libraries.

Libraries, Libraries and Again Libraries

- In the Clang/LLVM toolchain the two static analyzers are Clang-Check (a.k.a. Clang-SA) and Clang-Tidy.

- Clang-Check relies on a set of Clang modules to perform things like lexical analysis, parsing, semantic analysis, AST manipulation and the like.

- Clang-Tidy relies on the very same Clang modules plus some additional modules of Clang-Check itself (this is why Clang-Tidy can be considered a sort of superset of Clang-Check).

Libraries, Libraries and Again Libraries

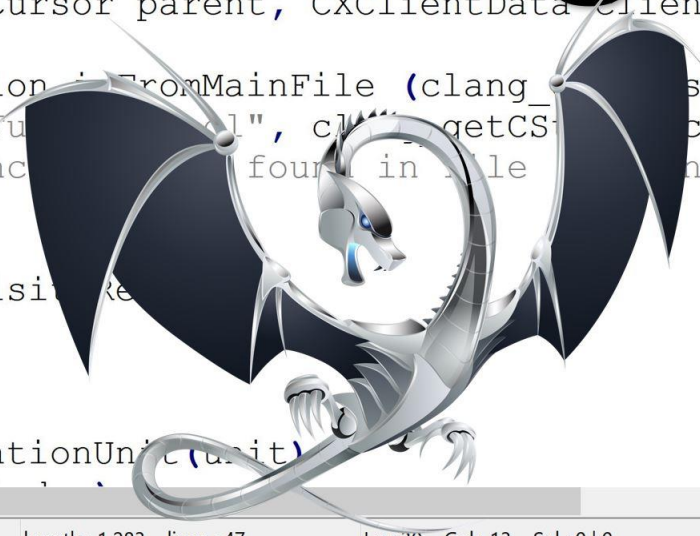
C:\naviserver-src\clang-visitor.cc - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

clang-visitor.cc

```
29
30  CXCursor cursor = clang_getTranslationUnitCursor(unit);
31  clang_visitChildren(
32    cursor,
33    [](CXCursor c, CXCursor parent, CXClientData client_data)
34    {
35      if (clang_Location
36        if (strcmp ("Fu
37          printf ("Func
38        }
39      }
40      return CXChildVisi
41    },
42    nullptr);
43
44  clang_disposeTranslationUni
```

libclang



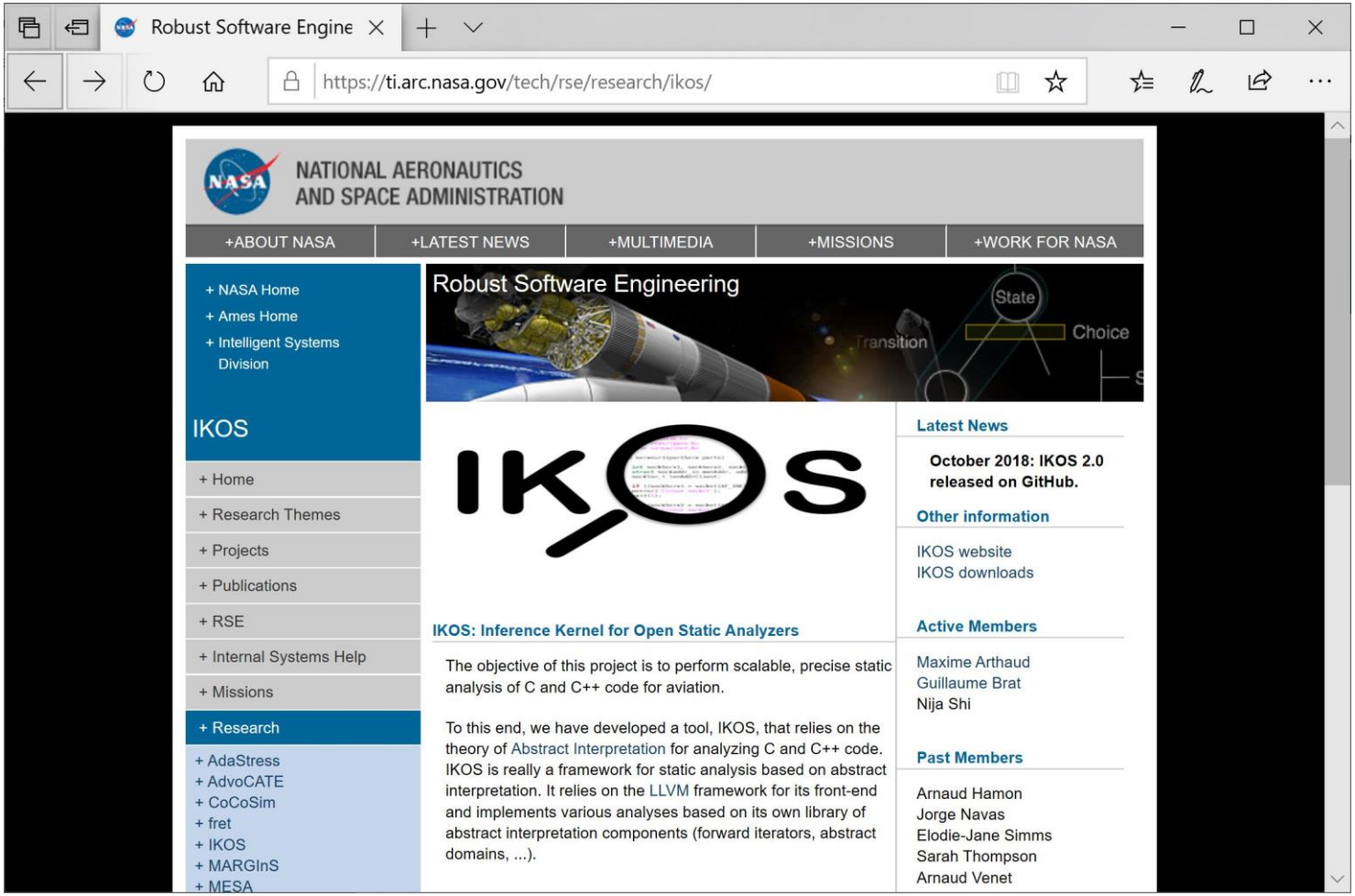
C++ source file | length : 1,283 lines : 47 | Ln : 30 Col : 13 Sel : 0 | 0 | Windows (CR LF) | UTF-8 | IN

Libraries, Libraries and Again Libraries

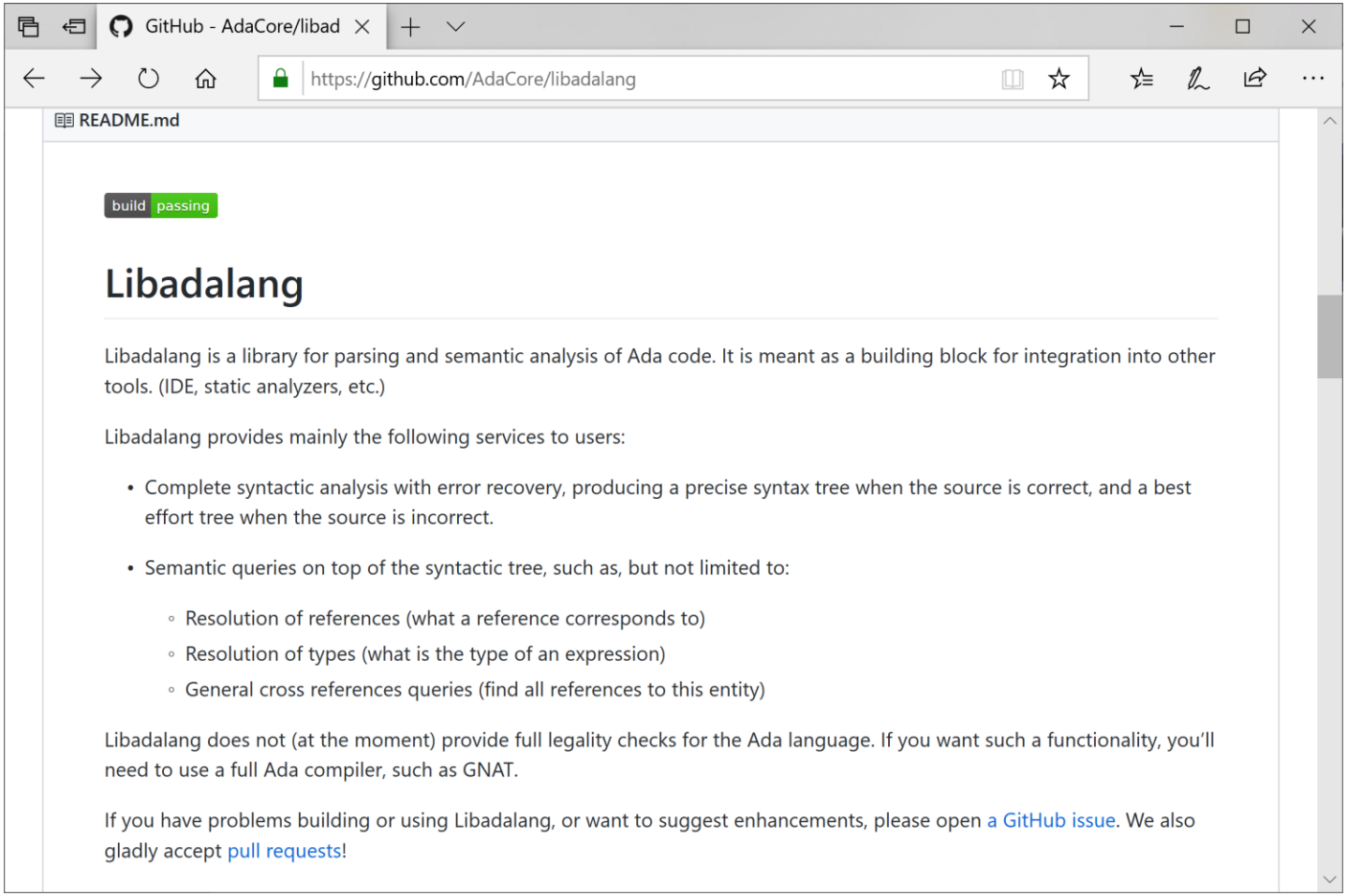
- “libclang” is nothing but a simple C API (with Python bindings) exposing Clang functionalities (i.e. modules) to external applications (deep / forgiving parsing);

- thanks to “libclang” also these third-party applications can use the very same modules/libraries of Clang (for instance they could parse a C program as efficiently as Clang does).

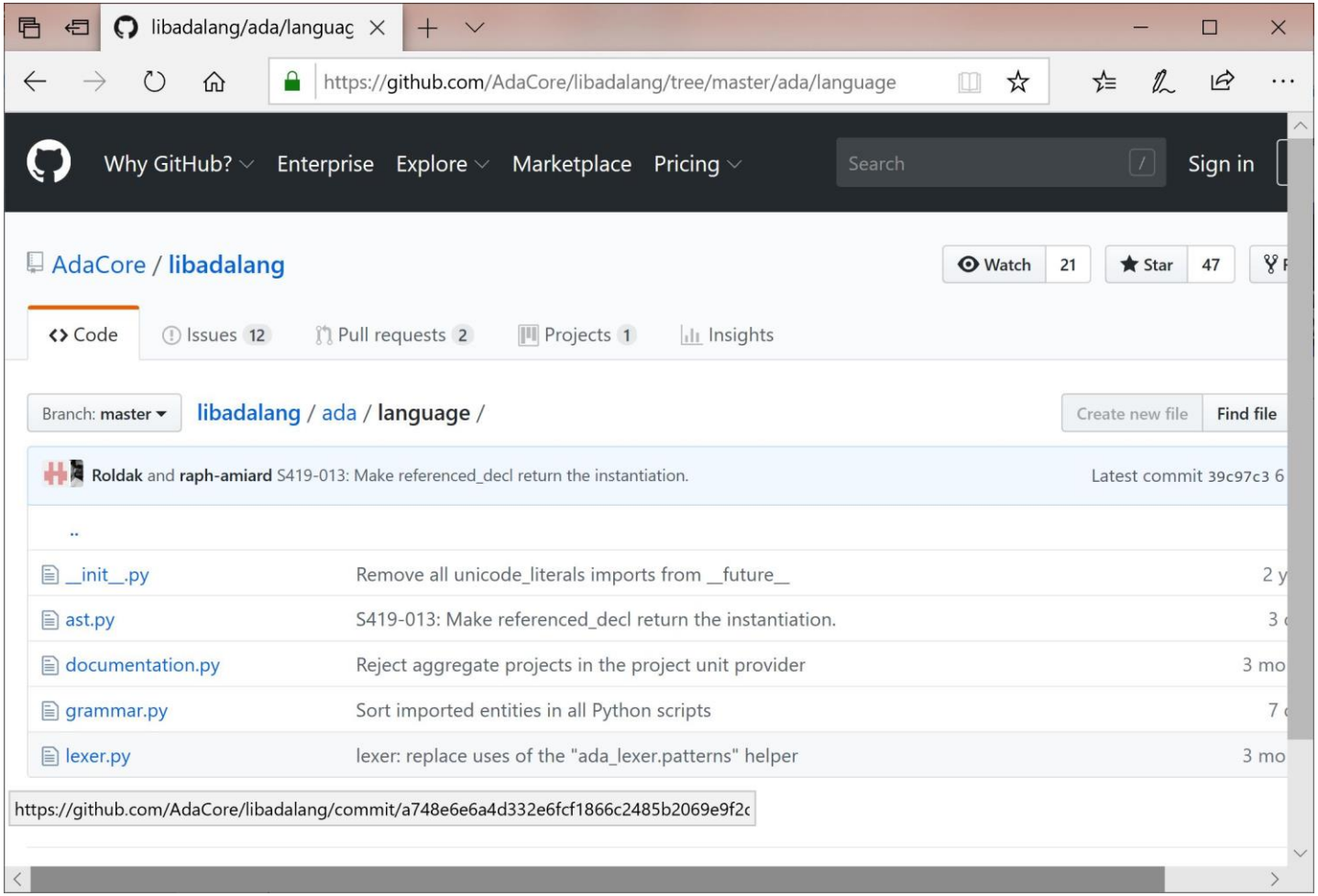
Libraries, Libraries and Again Libraries



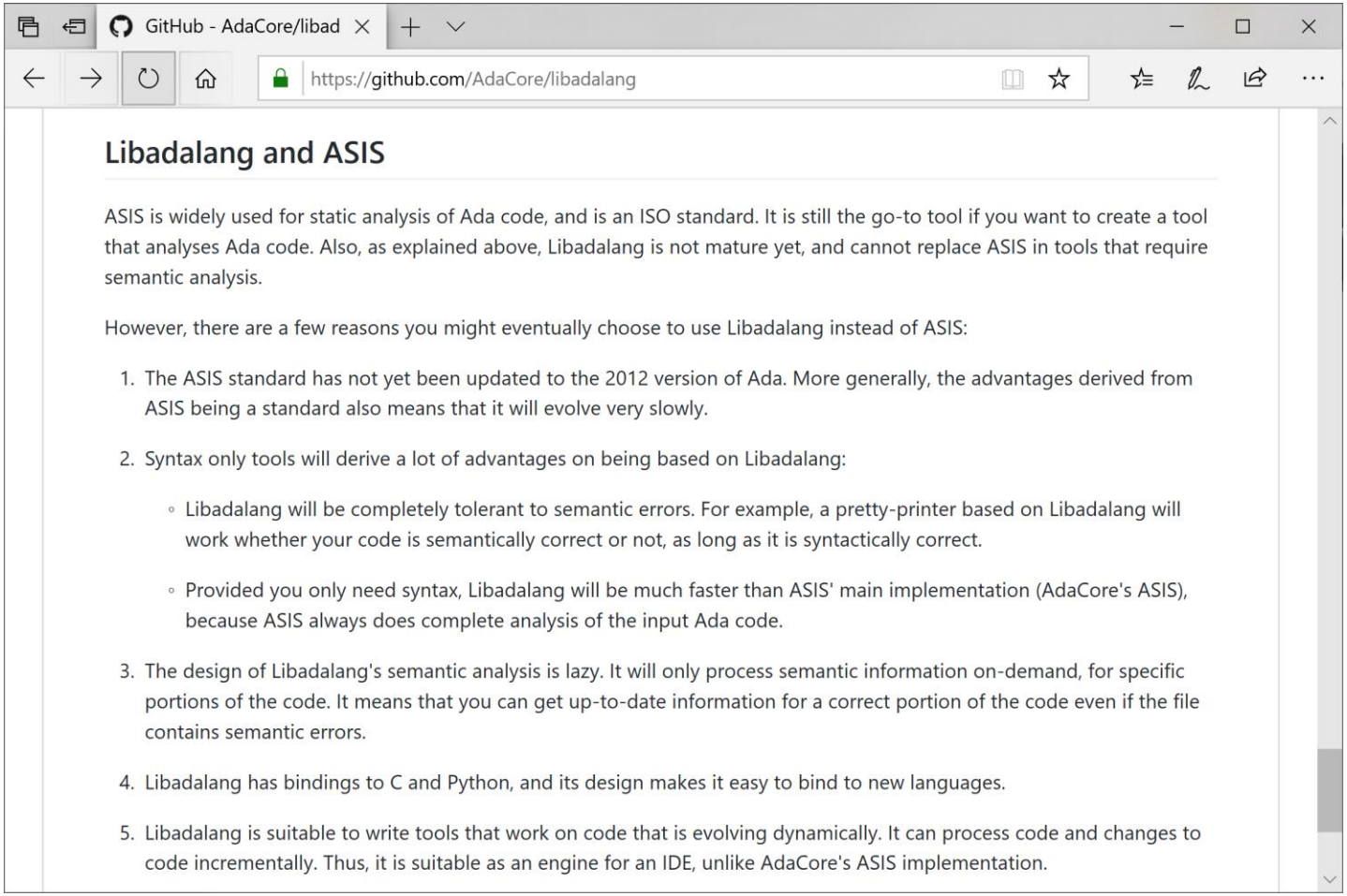
libadalang



libadalang



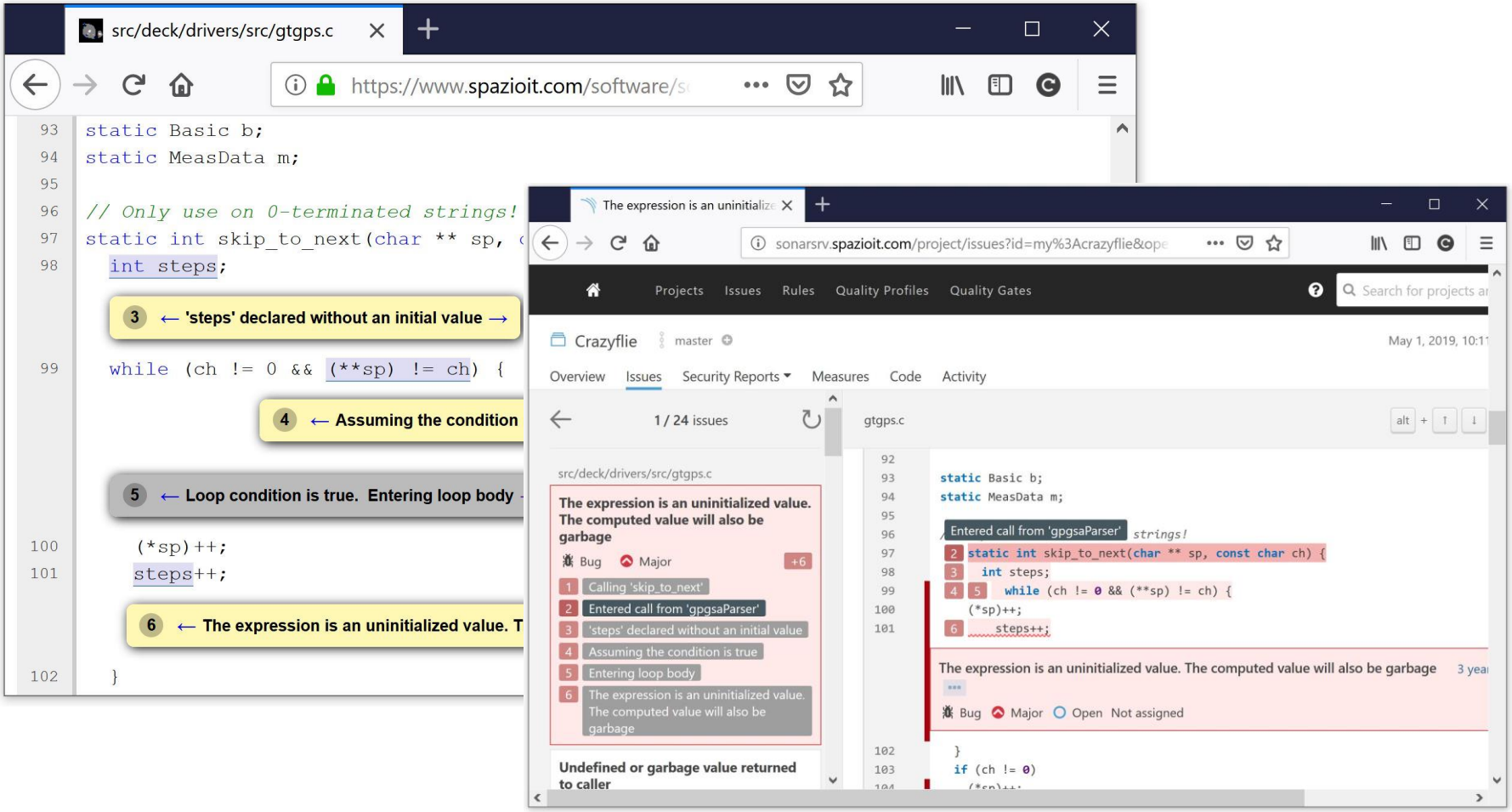
libadalang



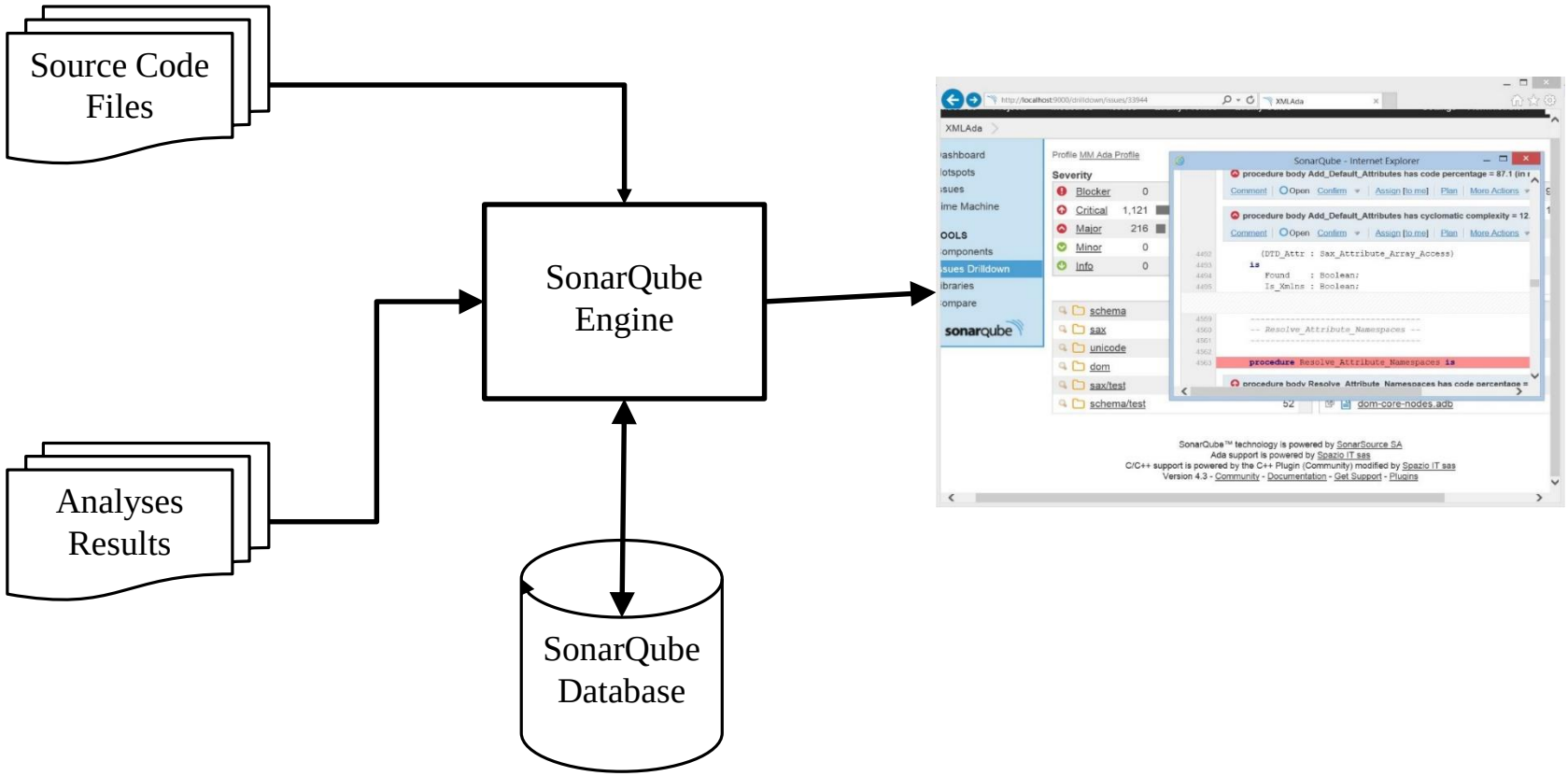
libadalang

- Interesting related projects:
  - **libadalang-tools** - Libadalang-based tools
  - **lal-checkers** - Libadalang-based code checking infrastructure
  - **ada_language_server** - prototype implementation of the Microsoft Language Server Protocol for Ada/SPARK
  - **langkit** - Language creation framework.

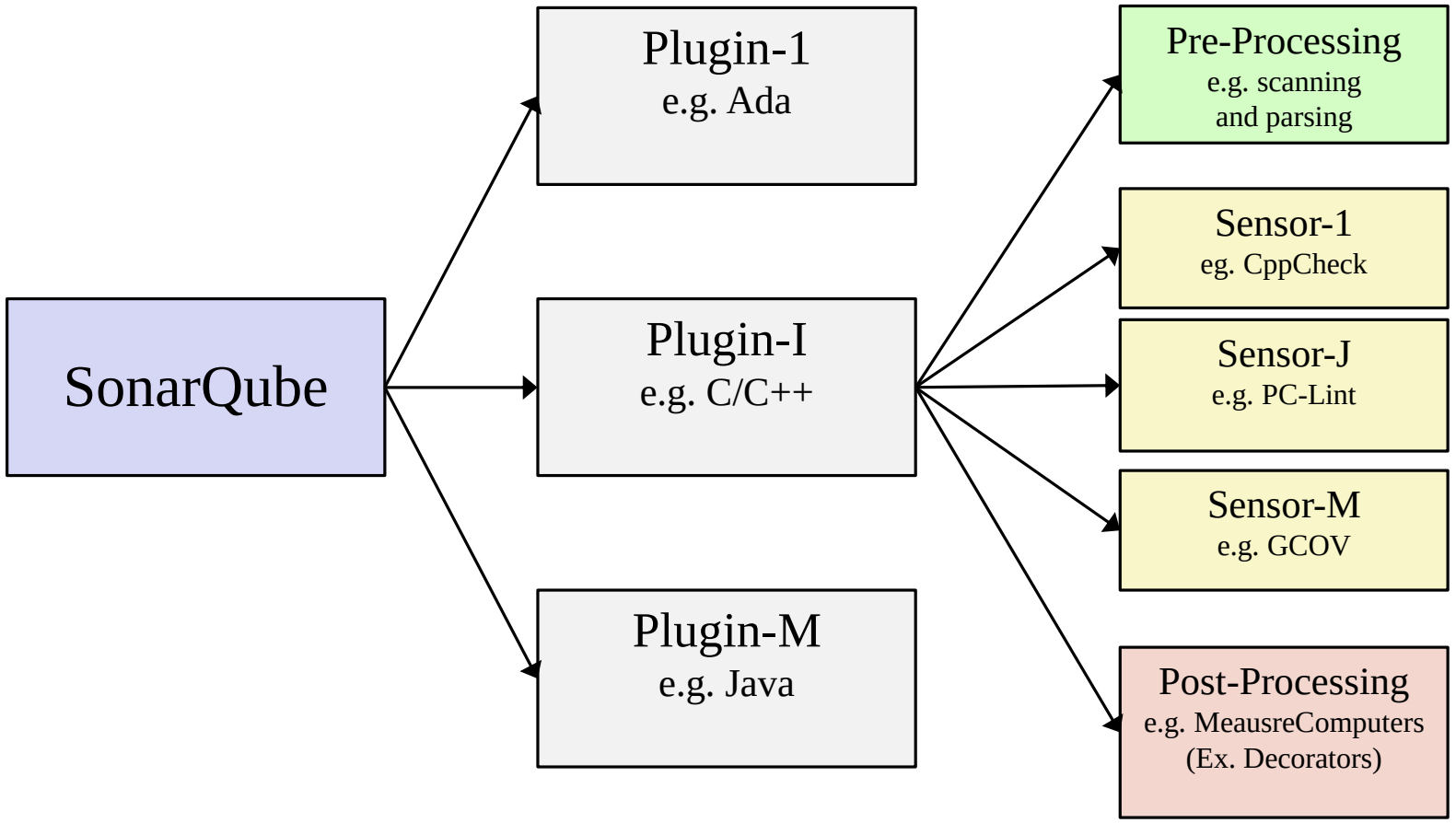
Clang / LLVM – SonarQube Integration



SonarQube – What is it?



SonarQube / Plugins / Sensors

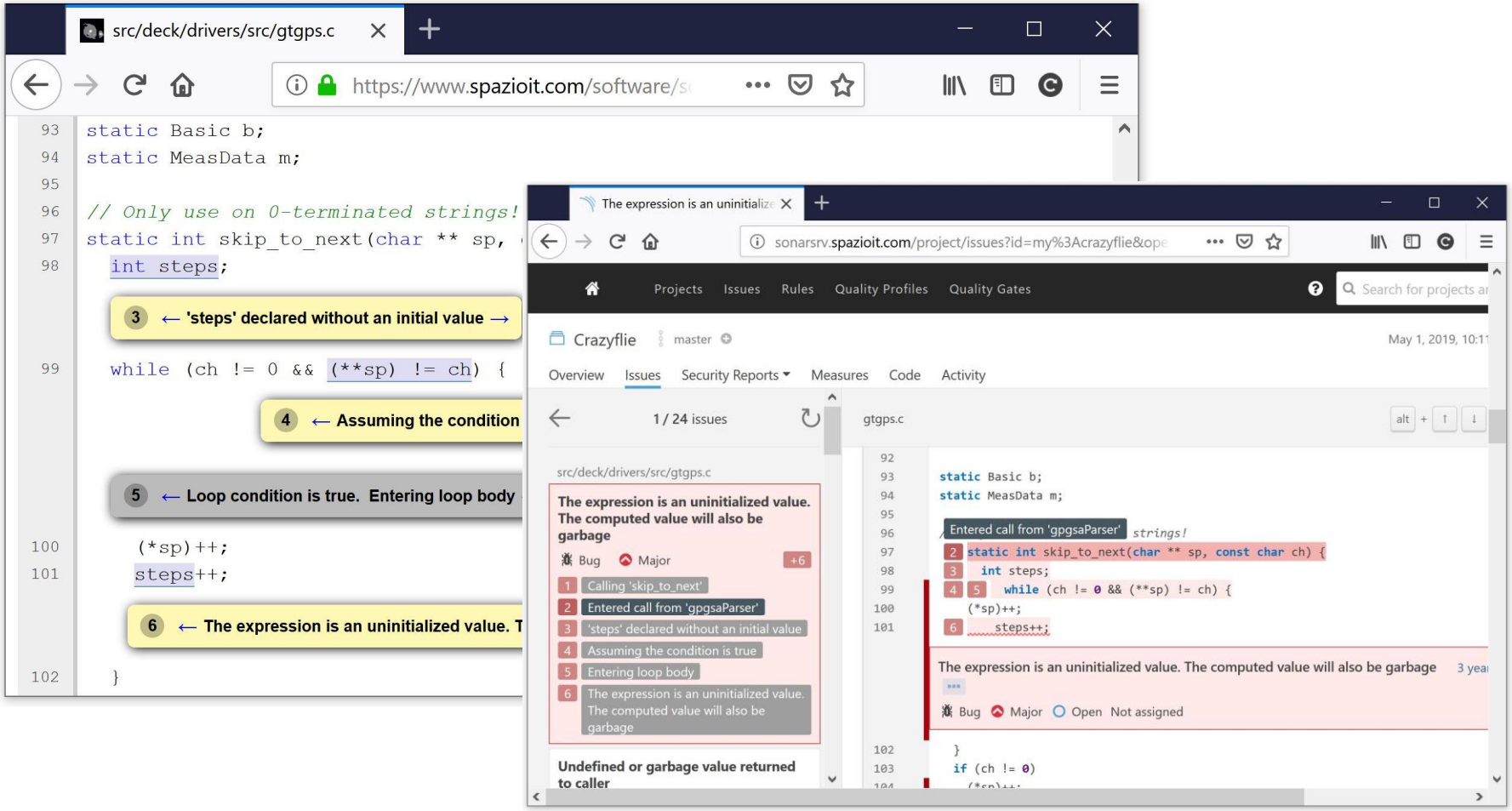


SonarQube C++ plugin (Community)

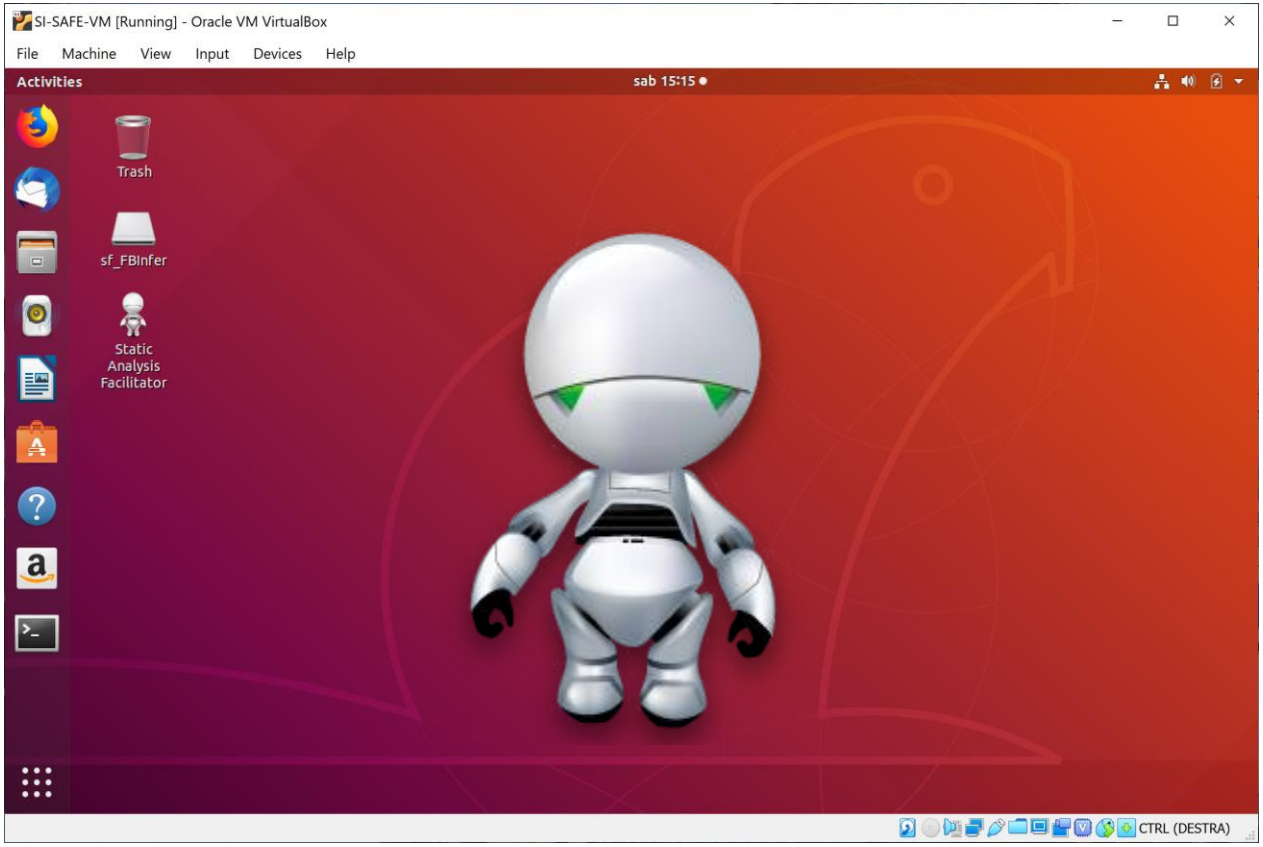
- Parser supporting C89, C99, C11, C++03, C++11, C++14 and C++17 standards
  - Microsoft extensions: C++/CLI, Attributed ATL
  - GNU extensions
  - CUDA extensions

- Sensors for static code analysis:
  - Cppcheck warnings support (http://cppcheck.sourceforge.net/)
  - GCC/G++ warnings support (https://gcc.gnu.org/)
  - Clang Static Analyzer support (https://clang-analyzer.llvm.org/)
  - Clang Tidy warnings support (http://clang.llvm.org/extra/clang-tidy/)
  - PC-Lint warnings support (http://www.gimpel.com/)
  - (…) many others

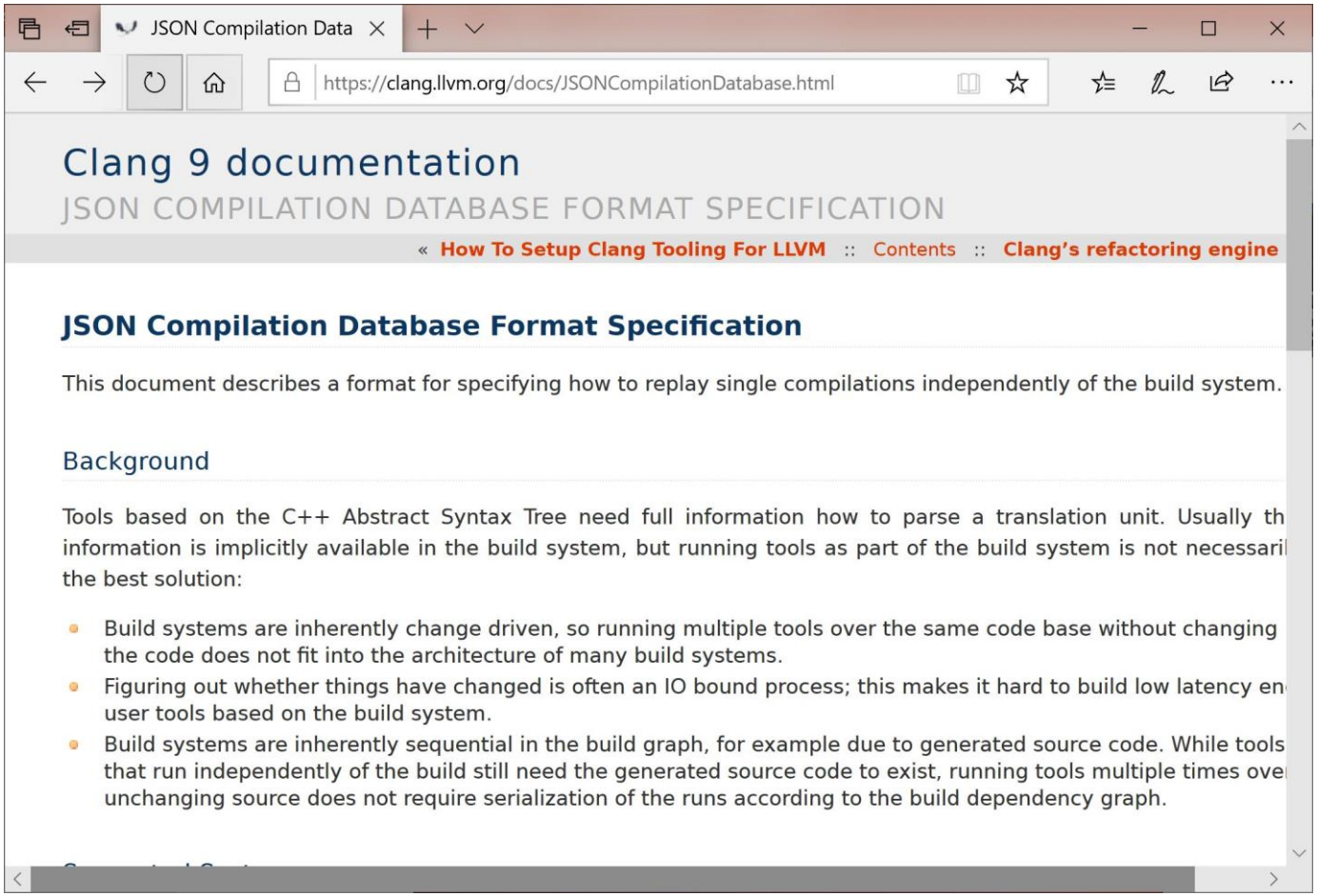
Clang / LLVM – SonarQube Integration



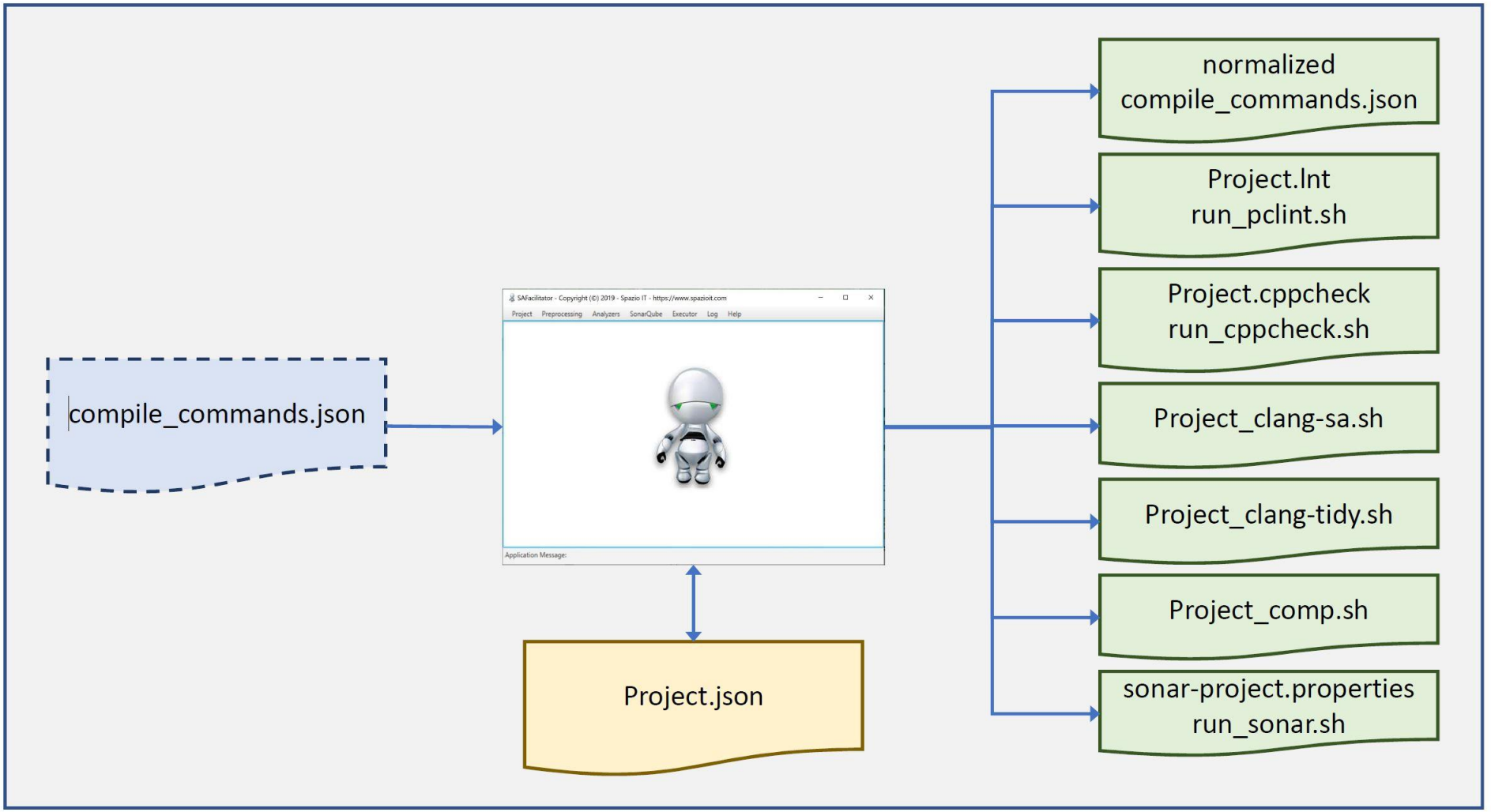
SAFe Toolset



SAFe Toolset



SAFe Toolset



SAFe Toolset

- The **SAFe Toolset** is an **Ubuntu Virtual Machine** containing **various open source tools** that can be used to perform **Software Verification** and **Validation**.

- In particular the current version (June 2019) of the SAFe VM contains:
  - **cppcheck** – v. 1.87 - http://cppcheck.sourceforge.net/ - a C/C++ static analyzer.
  - **Clang** – v. 9.0.0 - https://clang.llvm.org – the "new" compiler toolset from LLVM Foundation, with its **Clang-SA** and **Clang-Tidy** static analyzers.
  - **SonarQube** – v. 7.7. – https://www.sonarqube.org/ - a code quality platform used to show and manage the issues found by the static analyzers.

SAFe Toolset

- Optionally the SAFe VM may also contain:
  - **PC-Lint** (or PC-Lint Plus) – v. 9.0.0L - https://www.gimpel.com/ - but its license needs to be acquired from Gimpel.

- Apart from the static analyzers the SAFe VM contains also some (native and cross) build environments, that is:
  - **GNU GCC** Version 7.3.0 - https://gcc.gnu.org/gcc-7/ - Native
  - **Clang** Version 9.0.0 - - https://clang.llvm.org - Native and Cross (Multiplatforms – use the command "llc --version" to see the supported architectures).
  - **BCC2**: Bare-C Cross-Compiler System for LEON2/3/4 GCC 7.2.0 - https://www.gaisler.com/ - Cross.
  - **GNU Arm Embedded Toolchain** - v. 5-2016-q3 - https://launchpad.net/gcc-arm-embedded - Cross.

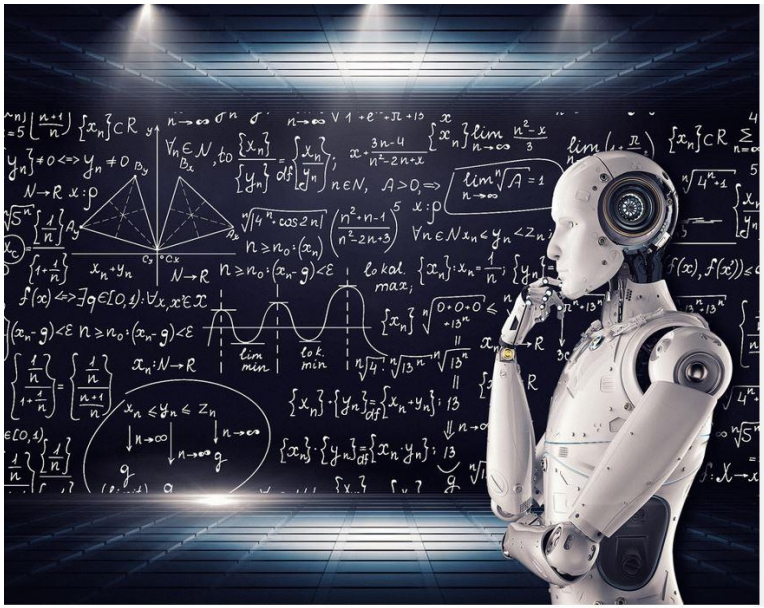
SAFe Toolset

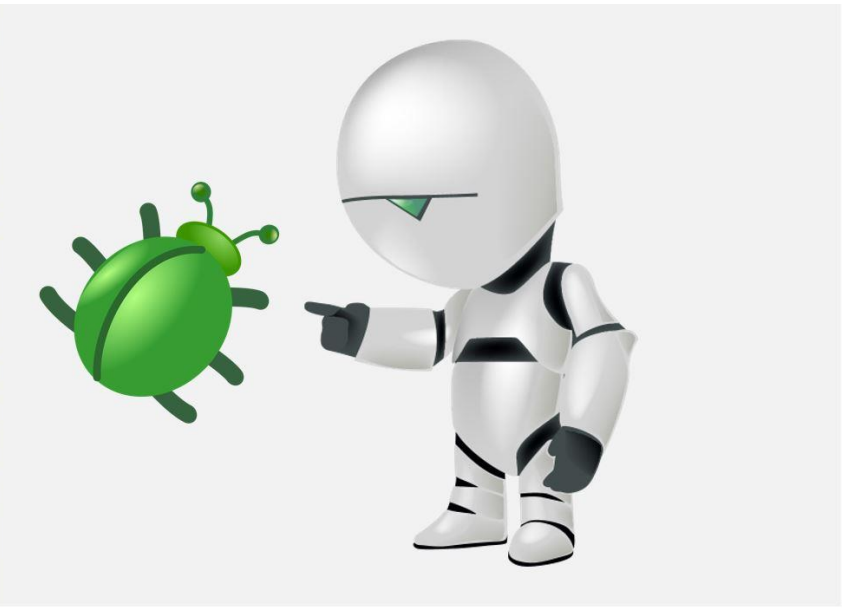
- Should a user need to work on a codebase not supported by the provided build environments, she would need to install the corresponding compilation toolchain.
- Additionally Spazio IT has complemented the SAFe Toolset with:
  - a specially **modified version of SonarQube** - https://www.sonarqube.org/ ;
  - a specially **modified version of the SonarQube C++ Community Plugin** - https://github.com/SonarOpenCommunity/sonar-cxx ;
  - the **SAFacilitator** – an application largely simplifying the static analyzers usage and the integration of their results into SonarQube – more info @ https://www.spazioit.com/pages_en/sol_inf_en/code_quality_en/safe-toolset/

SAFe Toolset

- The development of the SAFe Toolset has been funded by the European Space Agency Contract # RFP/3-15558/18/NL/FE/as.

Future/Current Activities





Future/Current Activities

- Spazio IT has just started working on Software Verification and Validation and Artificial Intelligence (especially Machine Learning). This research work is active on two complementary fronts:
  1. how to verify and validate AI software
  2. how to improve the “traditional” verification and validation activities with the adoption of AI techniques.
- Some new generations of static analyzers may be based on AI techniques.

Thank you for your time!