Quality of HAL libraries

T.O.M.A.S Team

life.augmented

Goal of this part

- Demonstrate basic information about code analysis techniques and its tools
- Present ST practices used for HAL and LL libraries quality verification

Program analysis

Program analysis is the process of analyzing the behavior of software programs to ensure quality attributes such as:

- correctness,
- reliability
- robustness,
- safety,

Program analysis can be performed:

- during code review
- without executing the program (static program analysis),
- during runtime (dynamic program analysis),
- combination of all above.

Static program analysis

Static program analysis is the analysis of computer software that is performed without executing a program.

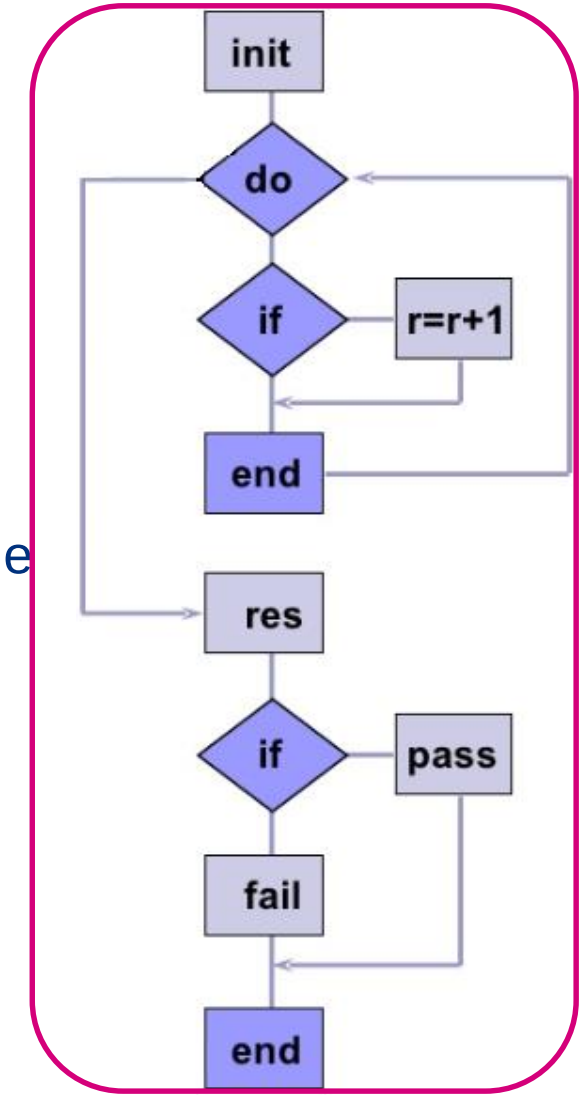
Two popular static program analysis methods:

- **Control-flow**: obtaining information about which functions can be called at various points during the execution of a program.
- **Data-flow**: gathering information about the values at each point of the program and how they change over time.

```
n := 0
read (x)
n := 1
while x > y do
  begin
    read (y)
    write( n*y)
    x := x - n
  end
```

Data flow anomaly: n is re-defined without being used

Data flow fault: y is used before it has been defined (first time around the loop)



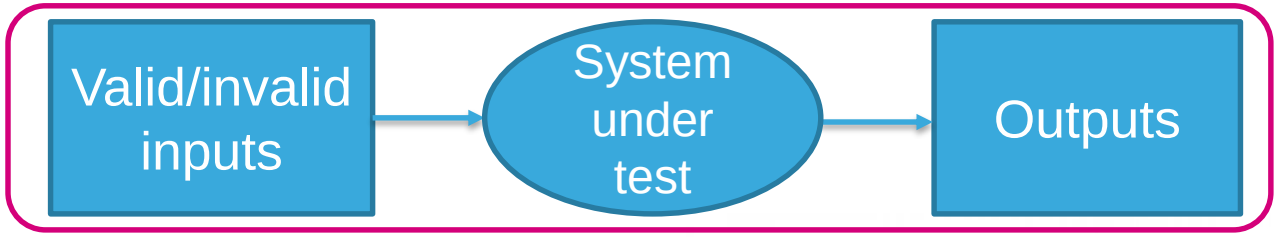
life.augmented

Dynamic program analysis

Dynamic program analysis is analysis performed on executing program.

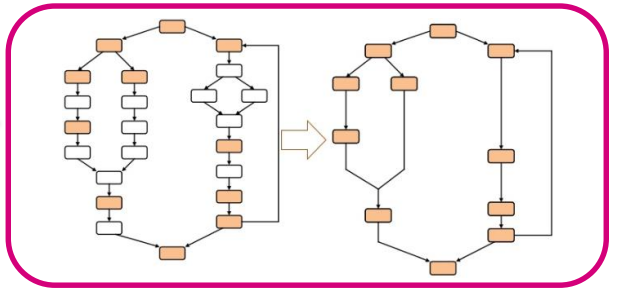
Popular dynamic program analysis methods:

- **Testing**: executing the program with a given input and evaluating its behavior and the produced output.



- **Monitoring**: records and logs different kinds of information about the program such as resource usage, events and interaction

- **Program slicing**: reducing the program to the minimum form that still produces the selected behavior.



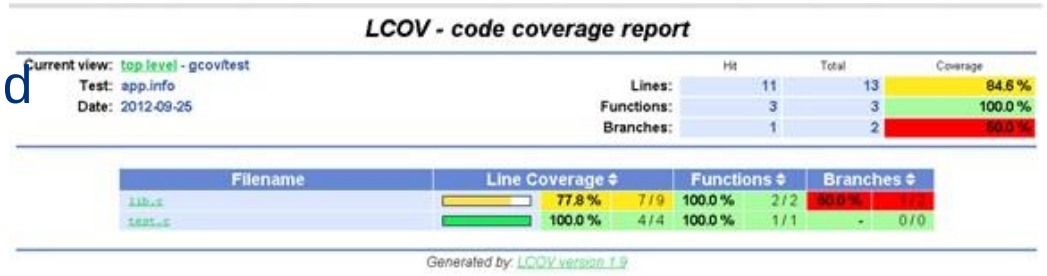
Program analysis tools

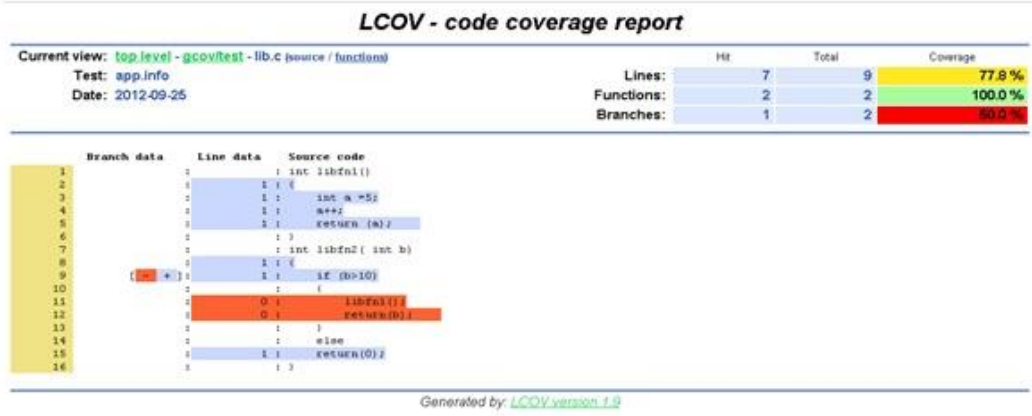
- Free software released under the terms of the GNU General Public License
  - **Splint** - tool for statically checking C programs for security vulnerabilities and coding mistakes.

Splint's output:

```
Variable c used before definition
Suspected infinite loop. No value used in loop test (c) is modified by test or loop body.
Assignment of int to char: c = getchar()
```

  - **Gcov** and **Gprof** - source code coverage analysis and statement-by-statement profiling tools.
    Gcov provides the following information:
    - How often each line of code executes
    - What lines of code are actually executed
    - How much computing time each section of code uses



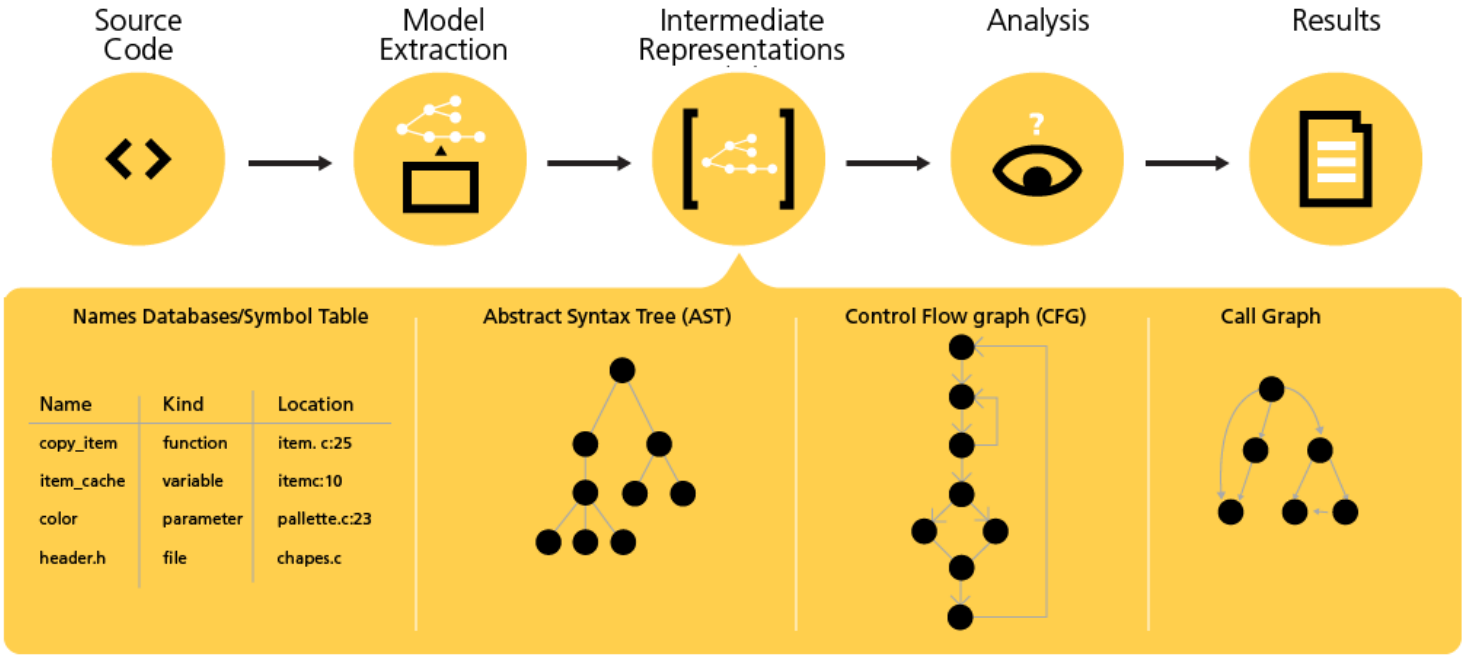


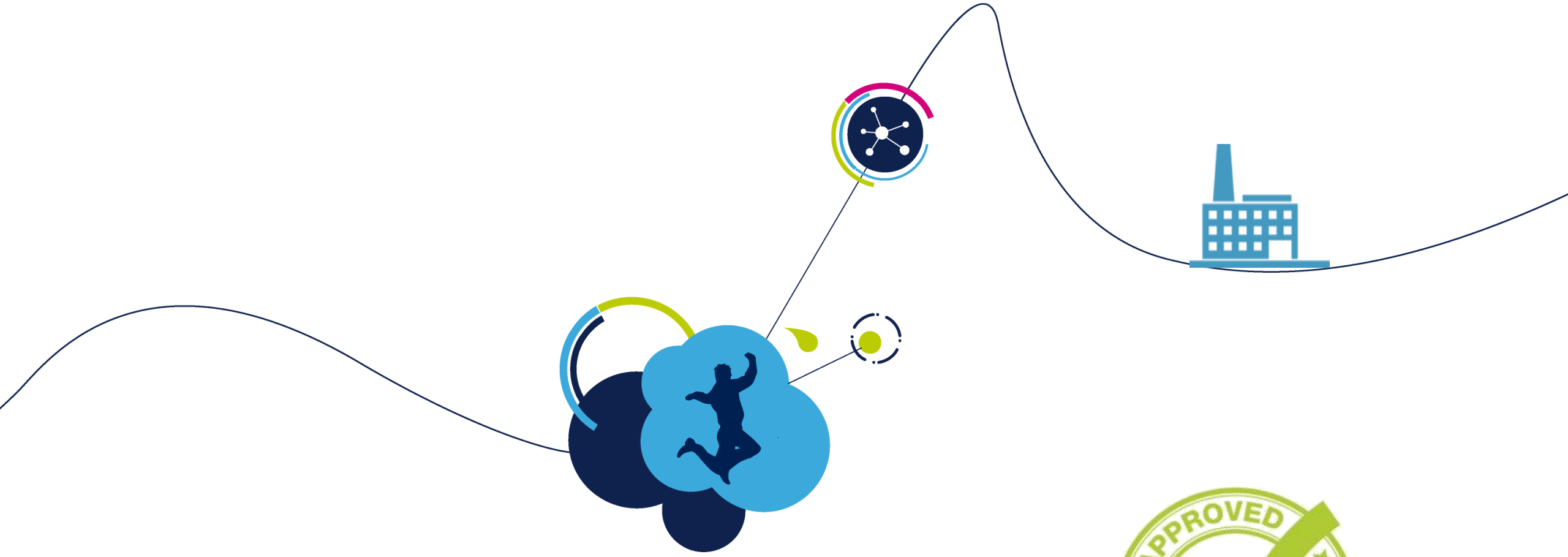
  - **LCOV** - a more user-friendly graphical visualization of the Gcov output.

Program analysis tools

- Commercial software
  - **CodeSonar -** static analysis tool from Grammatech, which identifies bugs that can result in system crashes, unexpected behavior, and security breaches.
    - Like a compiler, CodeSonar builds code using build environment, but instead of creating object code, CodeSonar creates an abstract model of entire program.
    - From the derived model, CodeSonar's symbolic execution engine explores program paths, reasoning about program variables and how they relate.

Tool used by ST to achieve high quality of Cube HAL





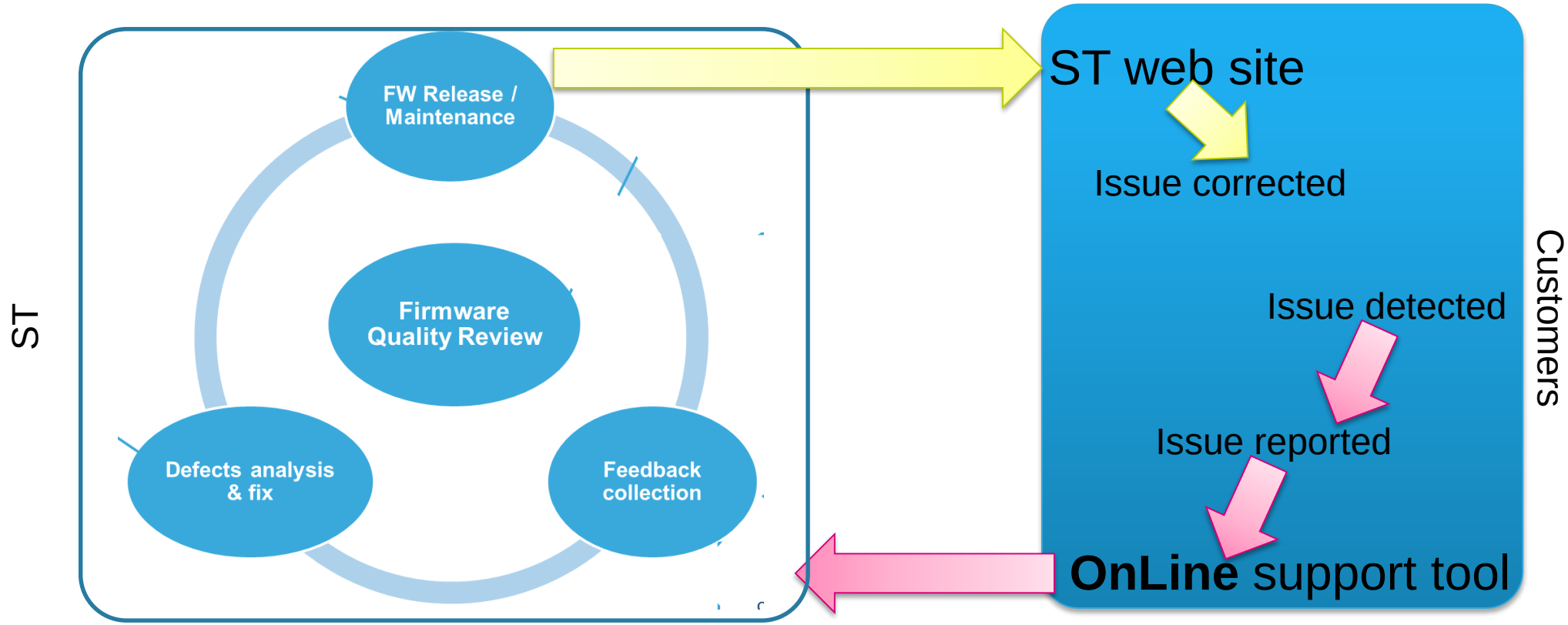
Quality of HAL libraries

life.augmented

STM32Cube High Quality Process

Maintenance Releases Policy

- The STM32Cube FW package is maintained regularly through:
  - Full release (marking x.y.**0**)
  - Patch release (marking x.y.z)
- The updater tool available with STM32CubeMX PC tool allows automatic notification and download of new STM32Cube release or patch

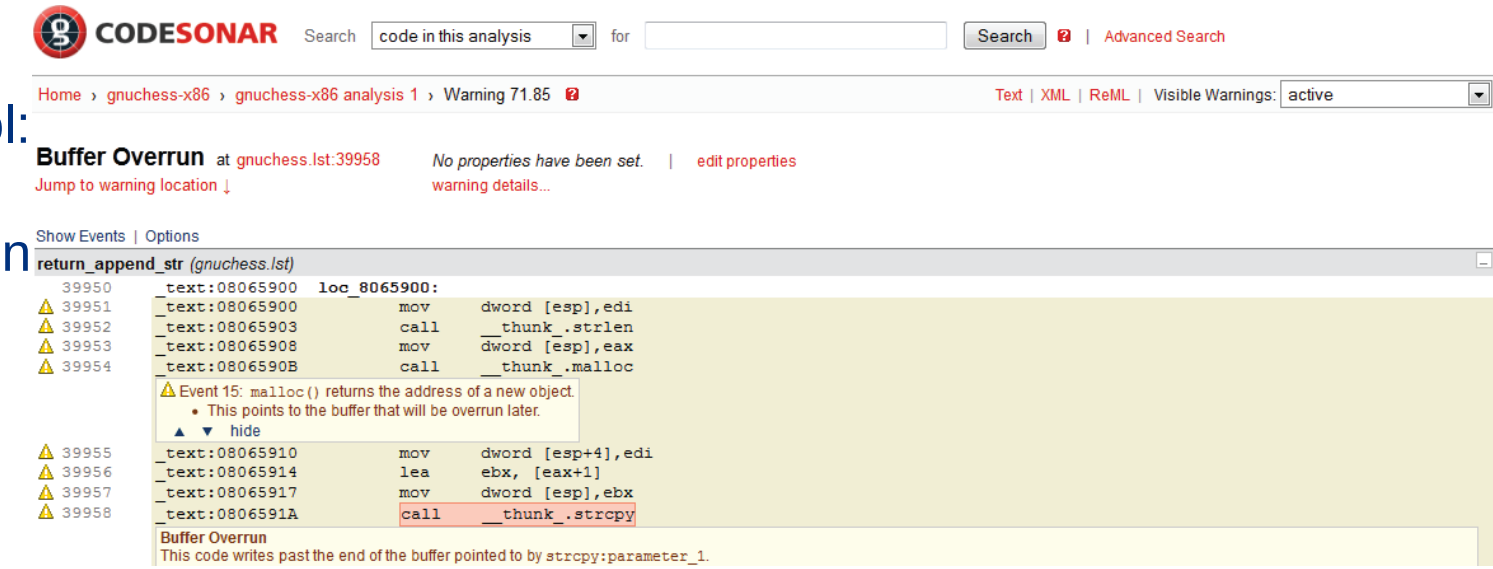


STM32Cube High Quality Process

Static code analysis

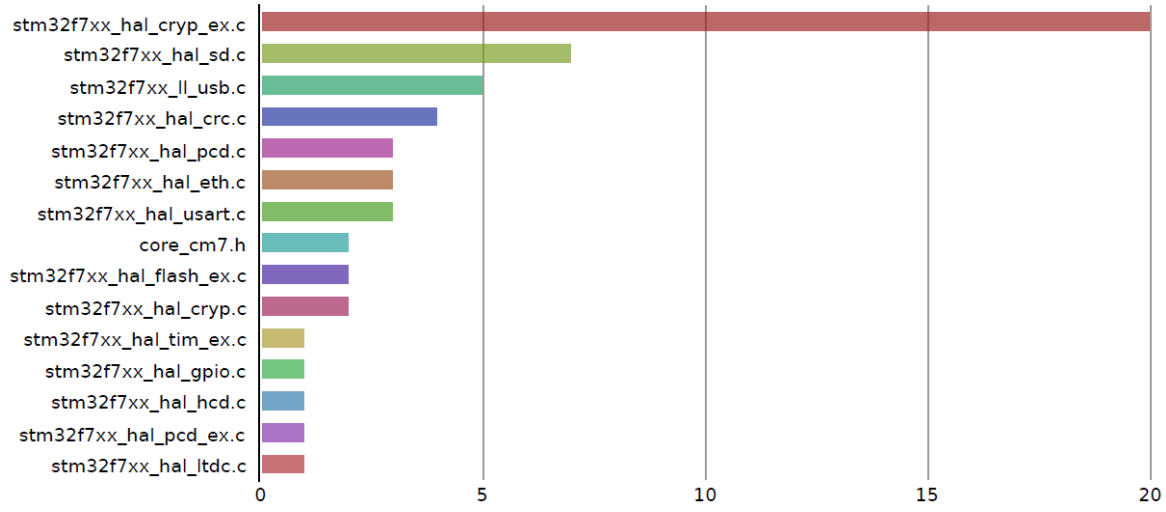
Static Code analyze

- HAL going through Code Analysis Tool: CodeSonar Industry reference tool identifying vulnerabilities at compilation time



- Report available on demand

Figure 4: Warnings by File



Example of the report

life.augmented

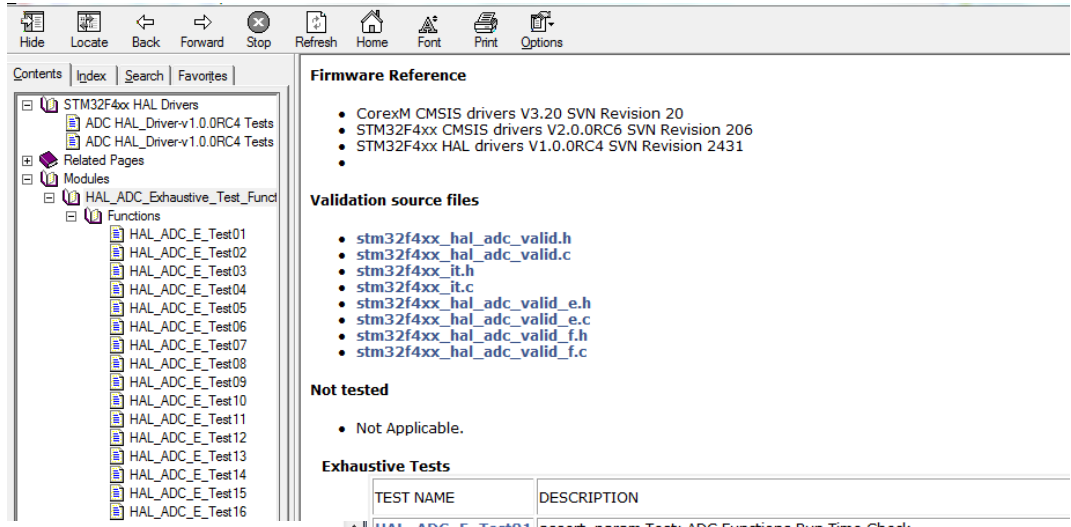
STM32Cube High Quality Process

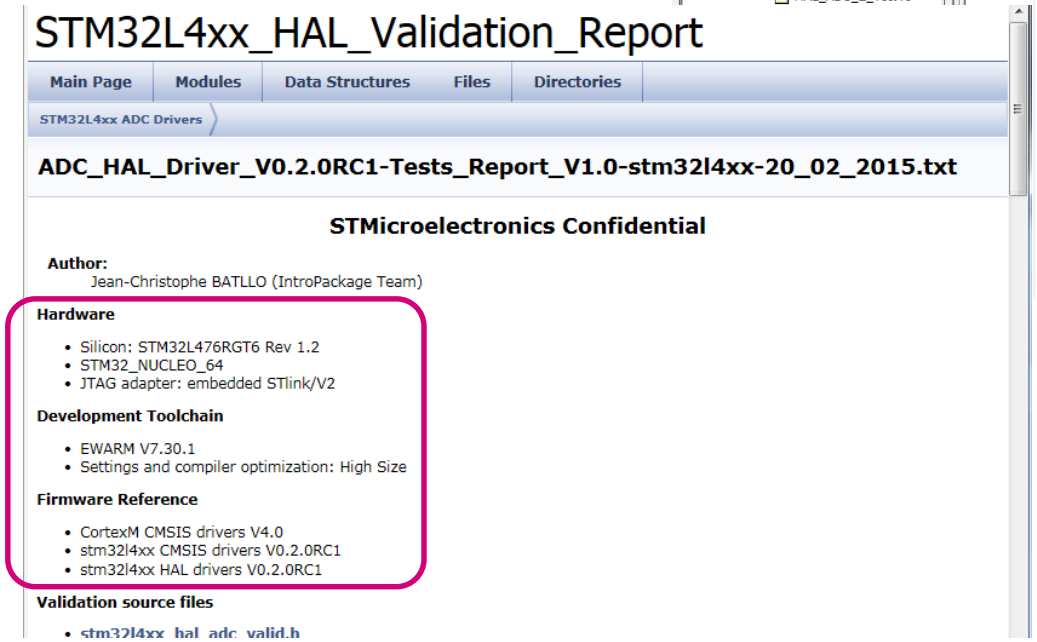
Dynamic code analysis and validation report

ST Dynamic validation test and Validation report on CUBE HAL

- Test are run by cube HAL i.e per sub family.
- One validation test and report per peripheral / function and package in Doxygen format
- Test are run (dynamic) in semi automatic way and use ST evaluation boards.

Example of the report





life.augmented

STM32Cube High Quality Process

MISRA C compliancy 1/2

STM32Cube HALs and ST Middleware C code compliant with MISRA-C

- Compliancy with a few exceptions listed and explained
- Check for MISRA-C 2004 is made using the IAR MISRA-C Checker (w/compiler: IAR C/C++ Compiler for ARM).
- Rules excluded from MISRA C Check (all Drivers):

Example of the report

MISRA-C 2004	Required/Advisory	Summary	Root Cause
1.1	Required	Compiler is configured to allow extensions - all code shall conform to ISO 9899 standard C, with no extensions permitted	IAR compiler extensions are enabled. This was allowed to support new CMSIS types.
5.1	Required	Identifiers (internal and external) shall not rely on significance of more than 31 characters	Some long parameters names are defined for code readability.
8.1	Required	Noprototype seen - functions shall always have prototype declarations and the prototype shall be visible at both the function definition	This rule is violated as there is no functions prototypes for __WFI and __WFE macros in the CMSIS layer.
10.1	Required	The value of an expression of integer type shall not be implicitly converted to a different underlying type.	Complexity
10.6	Required	A 'U' suffix shall be applied to all constants of 'unsigned' type	The "stdint.h" defined types are used to be CMSIS compliant.
11.2	Required	Conversionsshall not be performed between a pointer to object and any type otherthan an integral type, another pointer to object type or a pointer to void.	Needed when addressing memory mapped registers
11.3	Advisory	A cast should not be performed between a pointer type and an integral type.	Needed when addressing memory mapped registers
12.4	Required	The right-hand operand of a logical or | | operator shall not contain side effects.	This rule is broken in some drivers when conditional instruction "if" and a long expression is used, the solution is to split this instruction in several and small
14.7	Required	A function shall have a single point of exit at the end of the function.	In some case we need to exit from function w/ status of Timeout or error immediately and not waiting for the end of process function to update the

What have we learnt?

- ✓Demonstrate basic information about code analysis techniques and its tools
- ✓Present ST practices used for HAL and LL libraries quality verification

Further reading

More information can be found in the following documents (separately for each STM32 family):

- **STM32xx HAL library validation report**

  available on demand -> please contact your ST technical contact person

- **STM32xx HAL library MISRA C compliancy report**

  available on demand -> please contact your ST technical contact person

Enjoy!

www.st.com/mcu

life.augmented