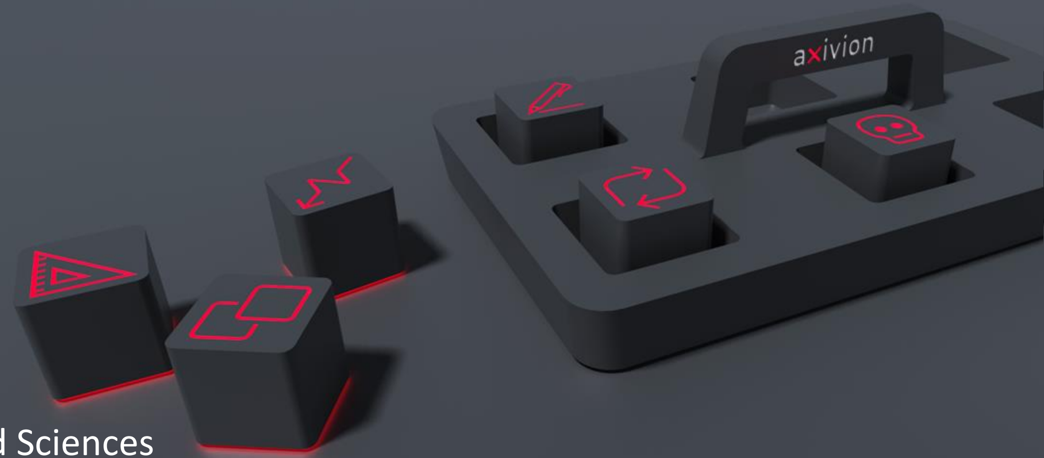
A Verified Low-Level Implementation of the Adaptive Exterior Light and Speed Control System



Sebastian Krings

Axivion GmbH, Niederrhein University of Applied Sciences

Philipp Körner, Jannik Dunkelau, Kristin Rutenkolk

University of Düsseldorf

axivion

stopping software erosion

axivion
stopping software erosion

This one is a little different than what we usually have at ABZ.

© 2021 Axivion GmbH

Differences to Other Case Studies

axivion
stopping software erosion

- We did not follow a (fully?) formal approach to the case study
- Instead:
    - Implemented ELS and SCS in low-level C
    - Used test-driven development
    - After some implementation, used model checking (on C!) for verification
- Attempted an approach closer to industry
    - No offense here!
    - Understand differences to help industrial uptake of FM
    - Provide a baseline to argue about advantages of FM in other case study realizations

© 2021 Axivion GmbH

Why Baselining?

axivion
stopping software erosion

- Some time ago, we read
  - Seven Myths of Formal Methods – Hall
  - Seven More Myths of Formal Methods – Bowen & Hinchey
- Both discuss myths regarding the costs of using FM and disprove them
- Yet, case studies implementing a system both formally and non-formally are
  - Done only seldomly
  - Often done either by teams of FM experts or non-experts
  - Tend to the extremes: all formal or not formal at all

axivion
stopping software erosion

- Test-driven development
  - For functional requirements
  - Realized using cmockery & ctest
- Restricted language Misra C
  - To avoid common pitfalls
  - Enforced by static analysis, to some extent dataflow and pointer analyses
- Model checking
  - For functional requirements
  - Directly on the C code
  - Realized using CBMC

© 2021 Axivion GmbH

axivion
stopping software erosion

- Development and style guidelines for C
- Introduced by the Motor Industry Software Reliability Association
- Improve safety and security by avoiding common pitfalls
- To some extend automatically checkable, others undecidable

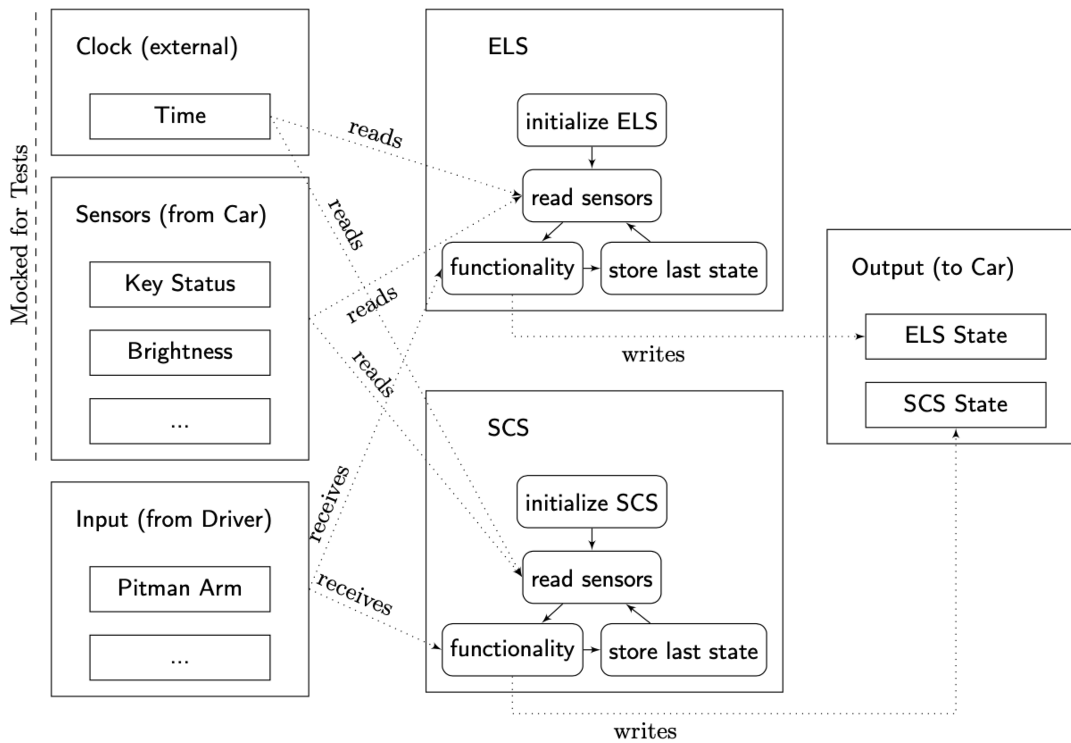
No verification of functional requirements here!

MISRA C:2012

Guidelines for the use of the C language in critical systems

March 2013

axivion
stopping software erosion



axivion
stopping software erosion

```
1  // ignition: key inserted + ignition on
2  sensor = update_sensors(sensor, sensorTime, 1000);
3  sensor = update_sensors(sensor, sensorBrightnessSensor, 500);
4  sensor = update_sensors(sensor, sensorKeyState, KeyInIgnitionOnPosition);
5  sensor = update_sensors(sensor, sensorEngineOn, 1);
6
7  mock_and_execute(sensor_states);
8
9  sensor = update_sensors(sensor, sensorTime, 2000);
10 pitman_vertical(pa_Downward5);
11 mock_and_execute(sensor_states);
12
13 assert_partial_state(blinkLeft, 100, blinkRight, 0);
14 pitman_vertical(pa_ud_Neutral);
15 sensor = update_sensors(sensor, sensorTime, 2000);
16 mock_and_execute(sensor);
17
18 pitman_vertical(pa_Upward7);
19
20 progress_time_partial(2000, 2499, blinkLeft, 100, blinkRight, 0);
21 progress_time_partial(2500, 2999, blinkLeft, 0, blinkRight, 0);
22
23 int i;
24 for (i = 3; i < 6; i++) {
25     progress_time_partial(i * 1000,       i * 1000 + 499,
26                           blinkLeft, 0, blinkRight, 100);
27     progress_time_partial(i * 1000 + 500, i * 1000 + 999,
28                           blinkLeft, 0, blinkRight, 0);
29 }
```

axivion
stopping software erosion

- Requirement: Whenever the low or high beam headlights are activated, the tail lights are activated, too.
- Translate into invariant
- Place assertion in C code, will be checked at runtime and during model checking

```
assert(implies(state.lowBeamLeft > 0,
               state.tailLampLeft > 0 &&
               state.tailLampRight > 0));
```

Model Checking using CBMC – Additional Assumptions

axivion
stopping software erosion

- CBMC unrolls transition system
- Needs to know cardinalities of enumerations, domain of variables, etc.
- Some are automatically detected, some depend on implementation (e.g. size of integers)
- Can be passed to CBMC so support it
- Again, this is a macro transparent to the C compiler

```
keyState ks = get_key_status();
__CPROVER_assume(ks == NoKeyInserted
              || ks == KeyInserted
              || ks == KeyInIgnitionOnPosition);
```

Model Checking - Example - CBMC Output

axivion
stopping software erosion

```
State 59 file light/light-impl.c line 242 function light_do_step thread 0
----------------------------------------------------
 ks=/*enum*/NoKeyInserted (00000000000000000000000000000000)

State 63 file light/light-impl.c line 242 function light_do_step thread 0
----------------------------------------------------
 ks=/*enum*/KeyInIgnitionOnPosition (00000000000000000000000000000010)

State 65 file light/light-impl.c line 244 function light_do_step thread 0
----------------------------------------------------
 engine_on=FALSE (00000000)

State 69 file light/light-impl.c line 244 function light_do_step thread 0
----------------------------------------------------
 engine_on=TRUE (00000001)
```

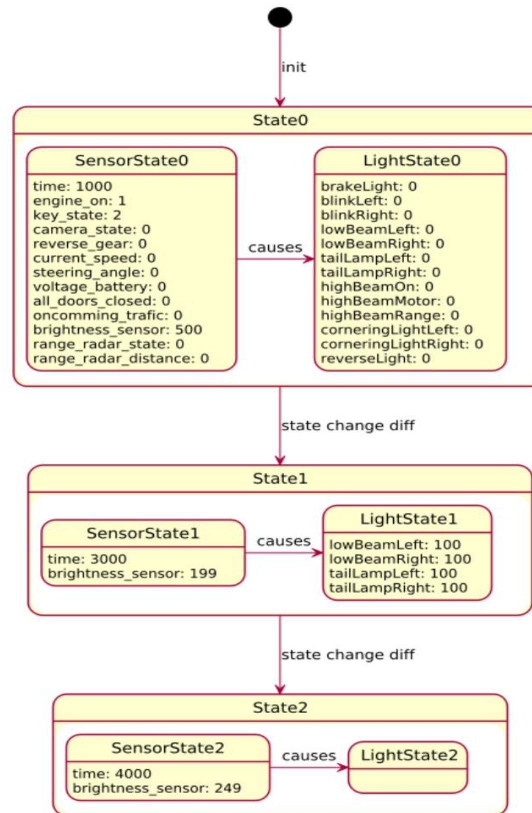
Model Checking – CBMC - Remarks

axivion
stopping software erosion

- CBMC found counterexamples, e.g., to the requirement ELS-22
- Output was barely readable, i.e., 52 state transitions represent two high level states after the initialization
- As a result, we wrote our own state graph visualization tool
- A simple C-style assertion would often have tripped the test case, CBMC adds thoroughness

Model Checking - Example – Ad-hoc Visualization

axion
stopping software erosion



© 2021 Axivion GmbH

axivion
stopping software erosion

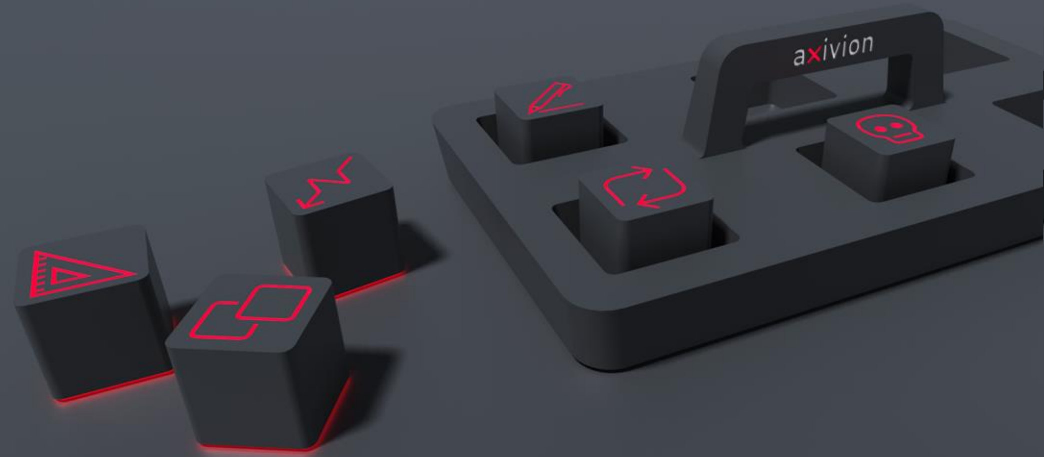
- Would like to follow up with an actual comparison of the different case studies
    - Discuss advantages and disadvantages of using FM
    - General lessons learned
- Idea: Make invariants from a formal model survive code generators and end up invariants in C code
    - Verify implementation again on actual system / in actual context
    - Spot errors in code generators

© 2021 Axivion GmbH

axivion
stopping software erosion

- We have
  - Implemented a low-level version of the case study
  - Used a language common in industry and broadly available infrastructure and techniques
  - Tried to stayed close to what might have happend in industry
  - Provided a baseline to compare more formal approaches to
- We suspect
  - More rigorous approaches will show advantages and disadvantages

Thank you for your attention!



Contact:
Sebastian Krings
Axivion GmbH
Nobelstraße 15
70569 Stuttgart

Tel: +49 711 6204378-78
E-Mail: krings@axivion.com
www.axivion.com

axivion
stopping software erosion