An Empirical Study On Practicality Of Specification Mining Algorithms On A Real-world Application

Mohammad Jafar Mashhadi

Hadi Hemmati

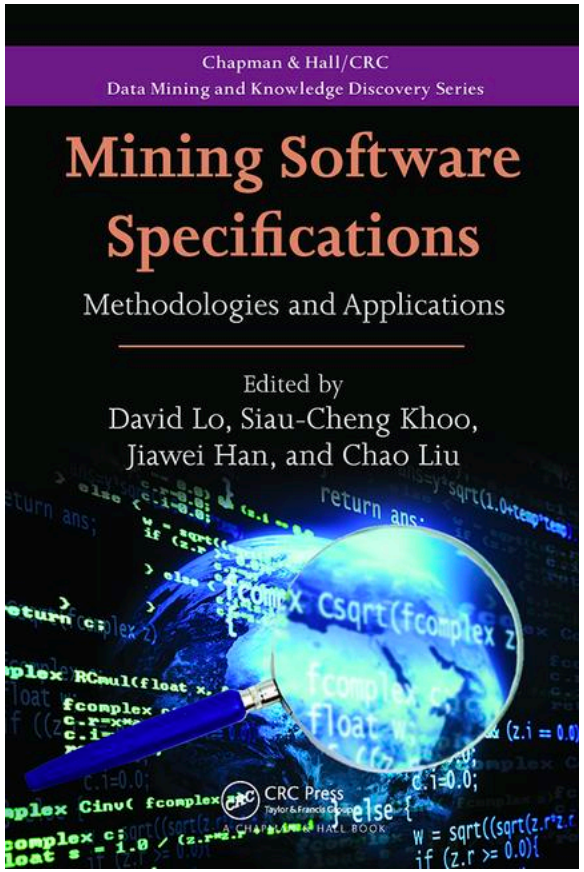
UNIVERSITY OF CALGARY

Natural Sciences and Engineering Research Council of Canada | Conseil de recherches en sciences naturelles et en génie du Canada

Canada

ICPC 2019 – Negative Results

Idea: Using Specification Mining for Debugging



Case Study: Autopilot software

MicroPilot®

UAVs

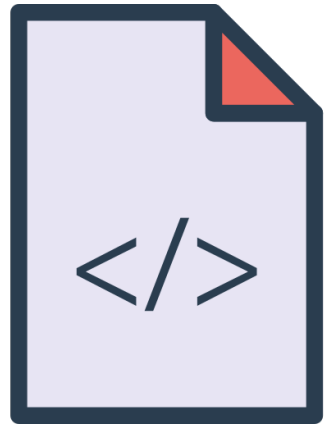
Fixed wing
Heli / Multirotor
VTOL
Crop cam drone
blimps

1000+

Customers

85

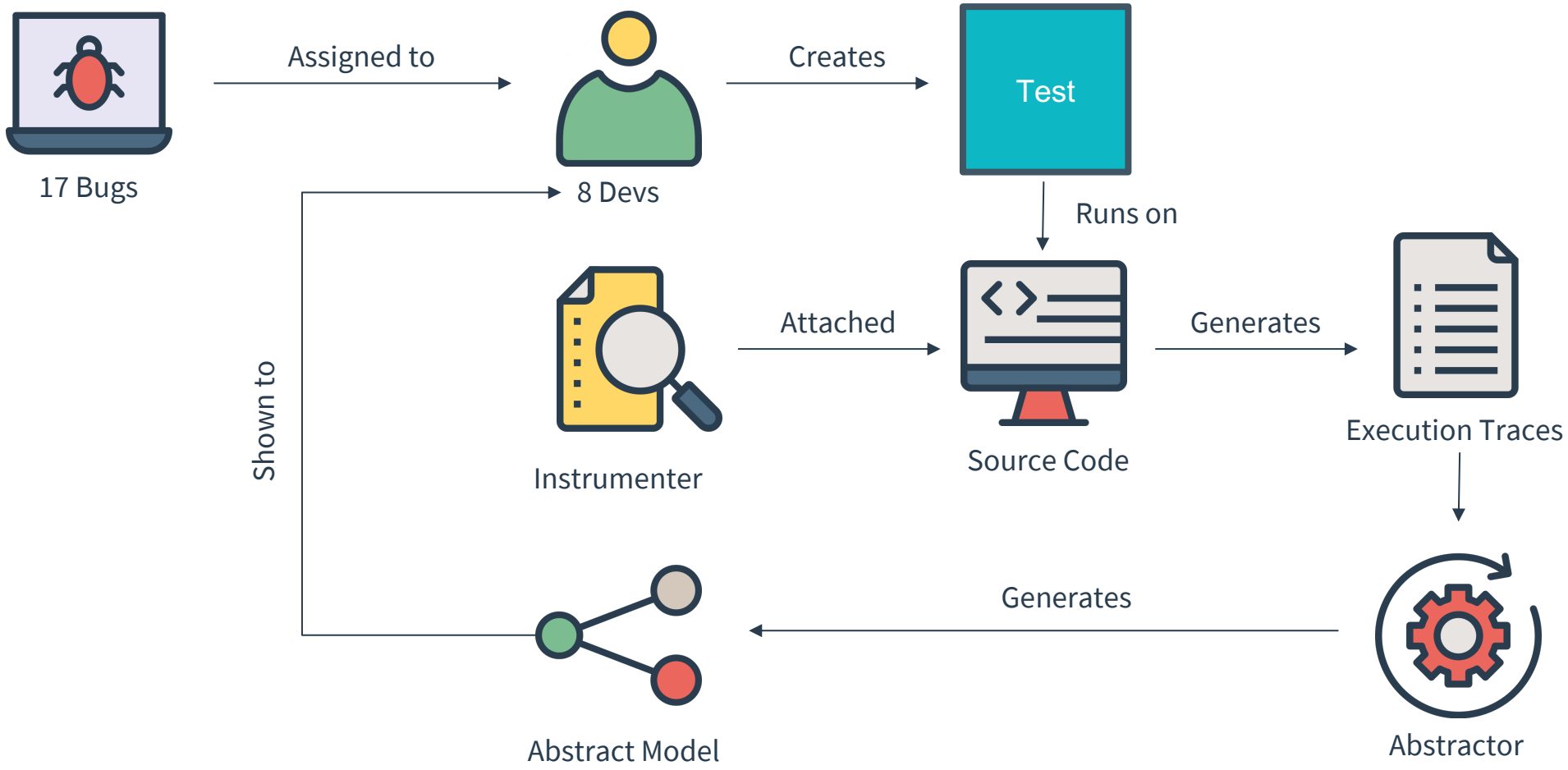
Countries



500K

LoC in C

Experiment Process Design – Big Picture



Experiment Process Design in This Study

Our focus:

- Instrumenter
- Abstractor

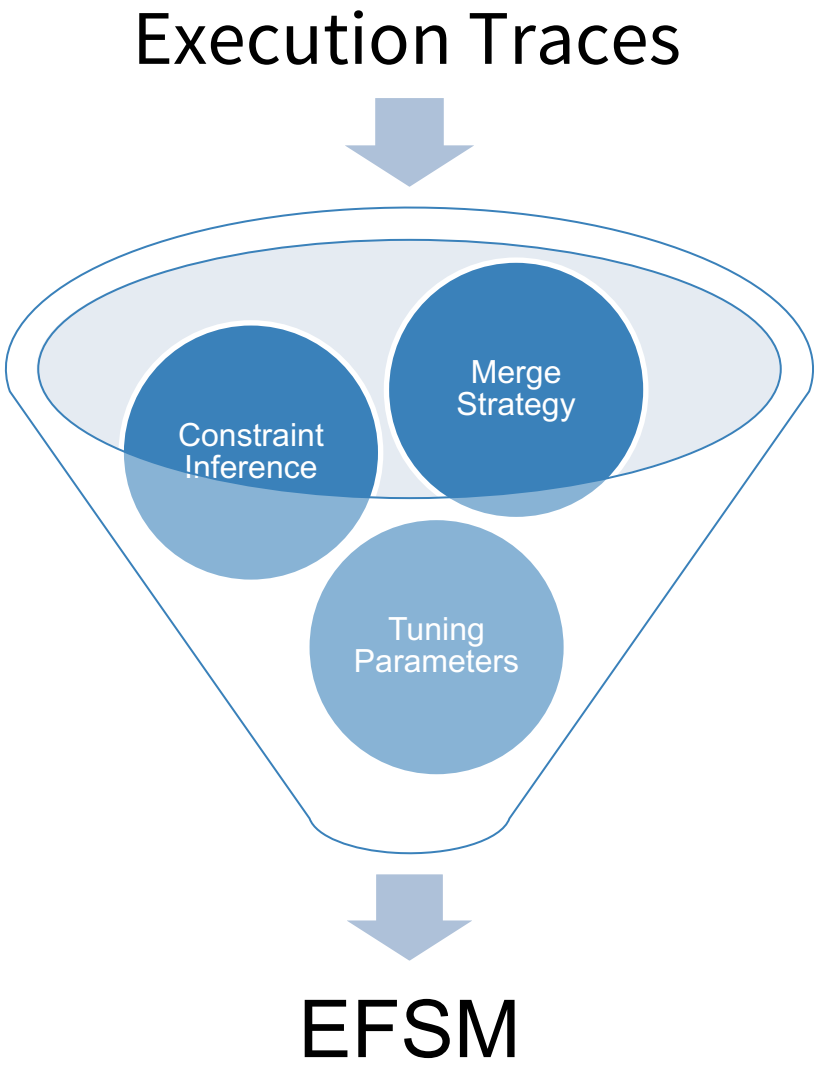
The challenges we faced while using existing tools and algorithms on 2 out of 17 bugs

Abstraction Tool: MINT

Modular and Configurable:

- Different Merge Strategies
  - k Tails
  - Gk Tail
  - EDSM (red-blue)
  - Exhaustive Merge
  - Noloops
- Careful Determinization
- Using Daikon

and more



Challenges

Instrumentation

Abstraction

Interpretation

Challenges

Instrumentation | Abstraction | Interpretation

- Too Many Events

100s of GBs of data

In a matter of

MINUTES

Challenges

Instrumentation

- Too Many Events
- Complex Variables

Abstraction

Interpretation

100+KB data structure

Nested

With *Pointers arrays, structs, buffers,...

Challenges

Instrumentation | Abstraction | Interpretation

- Too Many Events
- Complex Variables
- Irrelevant Variables

Not Relevant to The Bug

Challenges

Instrumentation

- Too Many Events
- Complex Variables
- Irrelevant Variables
- Boilerplate Code

Abstraction

Interpretation

```
template<async_overflow_policy OverflowPolicy = async_overflow_policy::block>
struct async_factory_impl
{
    template<typename Sink, typename... SinkArgs>
    static std::shared_ptr<async_logger> create(std::string logger_name, SinkArgs &&.
    {
        auto &registry_inst = details::registry::instance();

        // create global thread pool if not already exists..
        std::lock_guard<std::recursive_mutex> tp_lock(registry_inst.tp_mutex());
        auto tp = registry_inst.get_tp();
        if (tp == nullptr)
        {
            tp = std::make_shared<details::thread_pool>(details::default_async_q_size
            registry_inst.set_tp(tp);
        }

        auto sink = std::make_shared<Sink>(std::forward<SinkArgs>(args)...);
        auto new_logger = std::make_shared<async_logger>(std::move(logger_name), std:
std::move(tp), OverflowPolicy);
```

Challenges

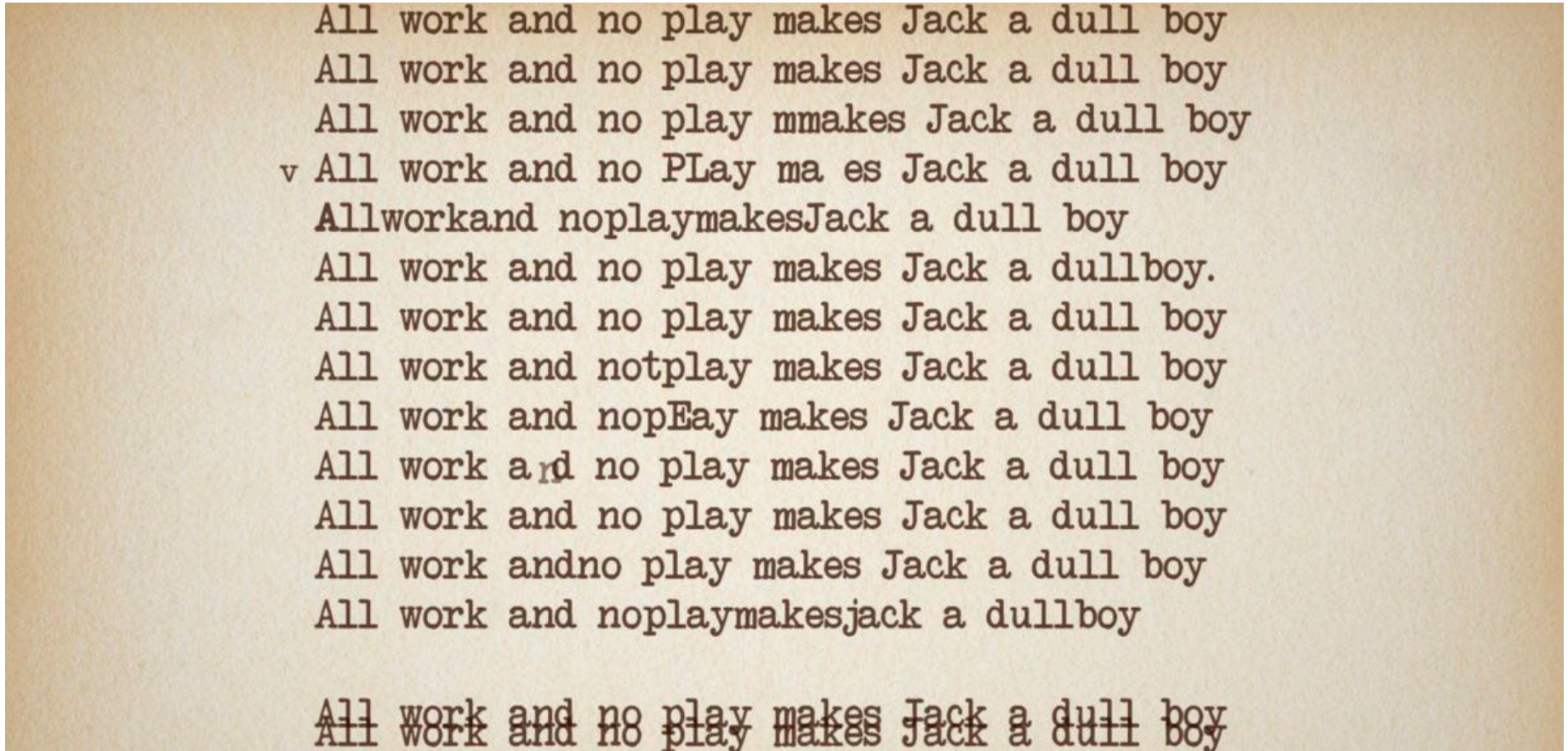
Instrumentation | Abstraction | Interpretation

- Too Many Events
- Complex Variables
- Irrelevant Variables
- Boilerplate Code

Solution: Making a whitelist of tracked functions and variables

Instrumentation | Abstraction | Interpretation

- Input Redundancy



Challenges

Instrumentation | Abstraction | Interpretation

- Input Redundancy

Removing adjacent identical events

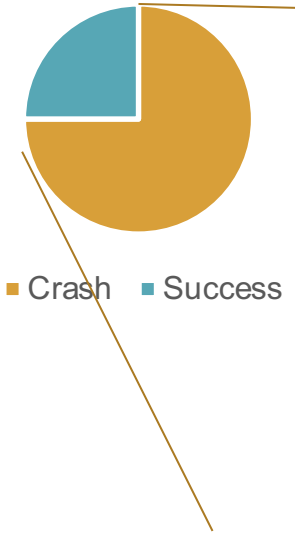
98% size reduction

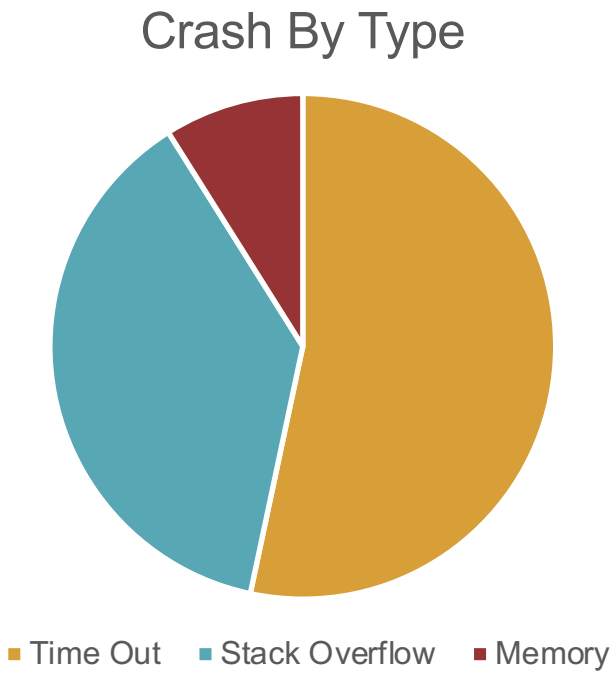
Challenges

Instrumentation | Abstraction | Interpretation

- Input Redundancy
- Crashing

Failure Type	Frequency
Time Out	40%
Stack Overflow	28.3%
Memory run out	6.7%





Challenges

Instrumentation | Abstraction | Interpretation

- Input Redundancy
- Crashing

Solution: Rewriting recursive functions, using iteration

Or optimising the implementation in general

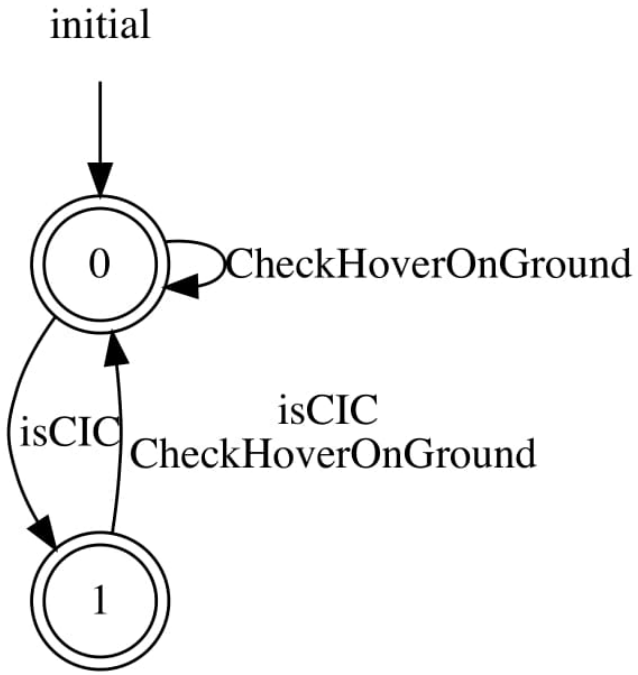
Challenges

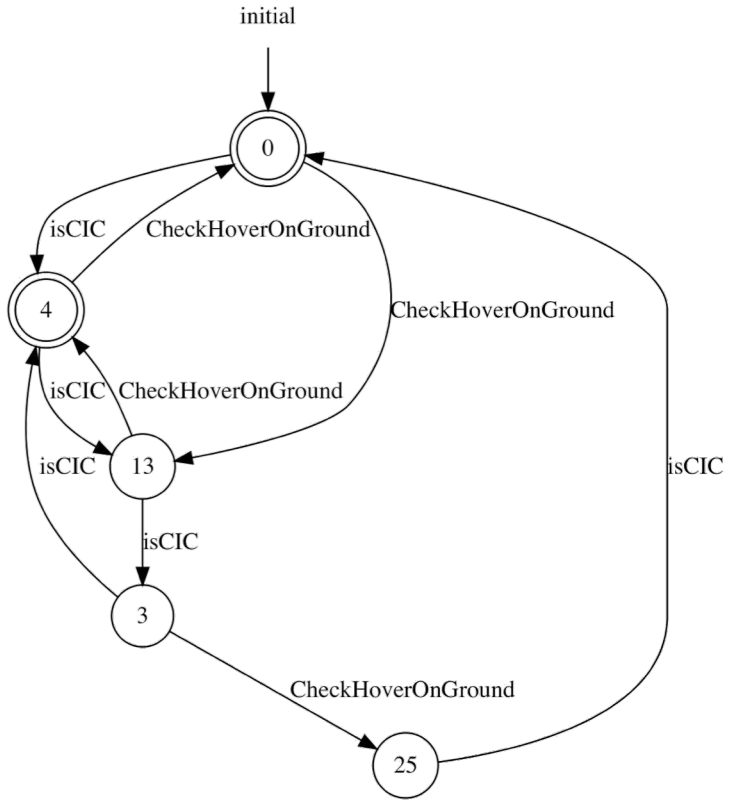
Instrumentation

Abstraction

Interpretation

- No Constraints
- Low Control on Granularity





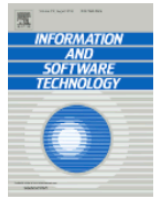
Keeping the human in the loop

ELSEVIER

Information and Software Technology

Available online 7 May 2019

In Press, Corrected Proof



Interactive semi-automated specification mining for debugging: An experience report

Mohammad Jafar Mashhadi [a], Taha R. Siddiqui [b], Hadi Hemmati [a], Howard Loewen [c]

Show more

https://doi.org/10.1016/j.infsof.2019.05.002

Abstract

Context

Specification mining techniques are typically used to extract the specification of a software in the absence of (up-to-date) specification documents. This is useful for

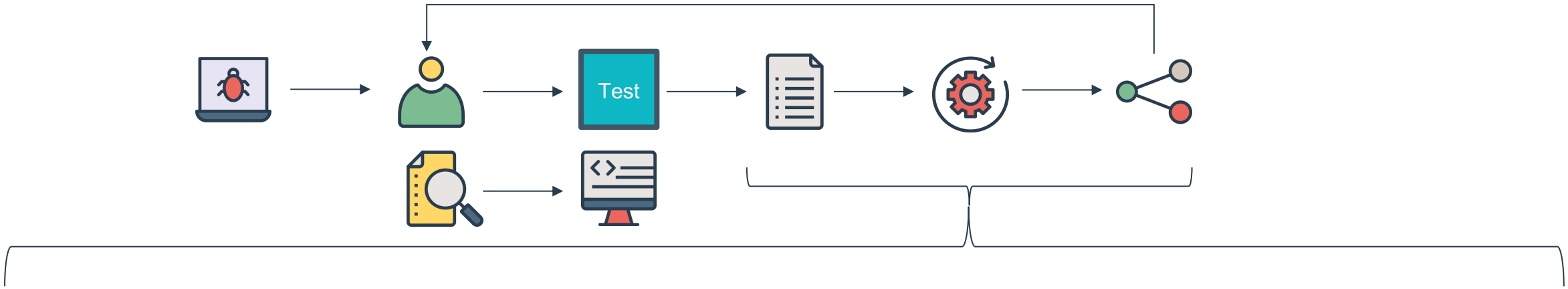
Take home messages

- Current specification mining algorithms are quite good, but their evaluation and application on large-scale and industrial settings can be improved.

- Pushing for full-automation can sometimes hurt practical applicability

Summary

MicroPilot®



Instrumentation

- Too Many Events
- Complex Variables
- Irrelevant Variables
- Boilerplate Code

Abstraction

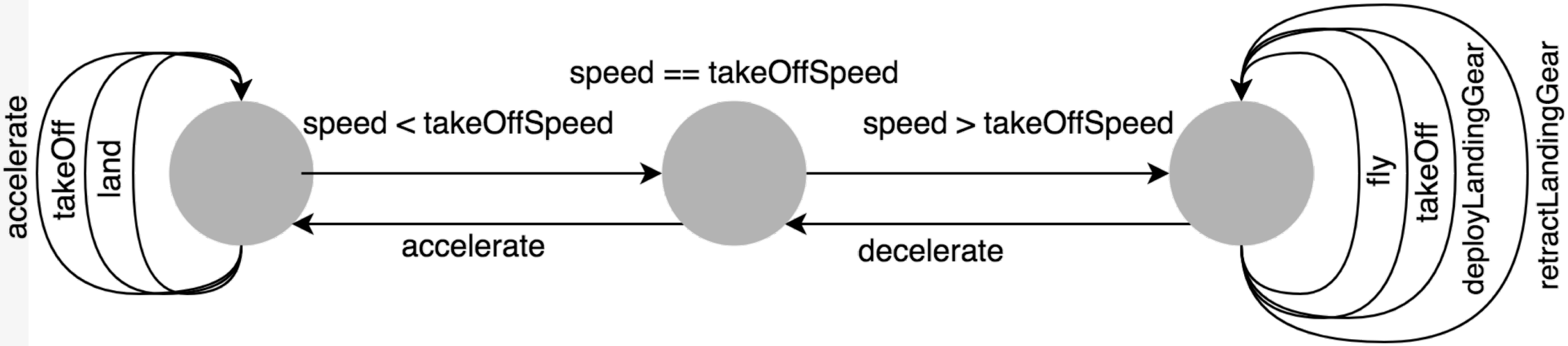
- Input Redundancy
- Crashing
  - Timing Out
  - Stack Overflow
  - Out of Memory

Interpretation

- No Constraints
- Low Control on Granularity

Which model works better?

Extended Finite State Machines



Why EFSMs are good?

1. The big picture
   - They are Intuitive and easy to use and show the whole behaviour in a glance
2. They show states!
   The software in our study is state-based
3. Constraints
   They contain the data associations
4. The algorithms and tools
   There are multiple algorithms available for state-machine inference in addition to their implementation