Combining Requirement Mining, Software Model Checking and Simulation-Based Verification for Industrial Automotive Systems

2016/10/06

Tomoya Yamaguchi and Tomoyuki Kaga

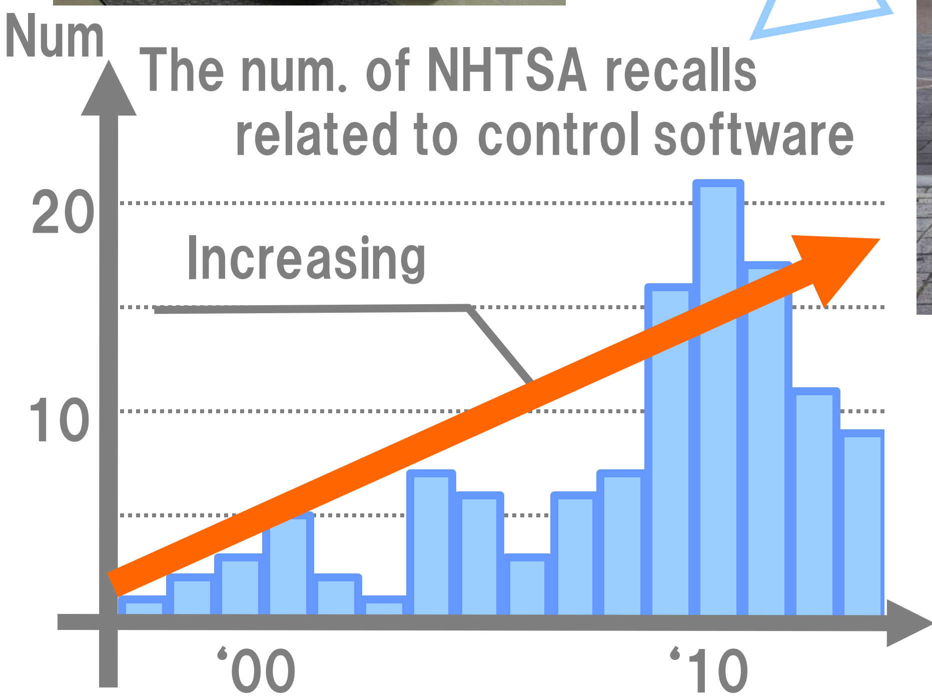
TOYOTA MOTOR CORPORATION

Alexandre Donzé and Sanjit A. Seshia

University of California, Berkeley

TOYOTA V&V Perspective

Automobile system becomes more complex and larger in scale.



Automotive system is really

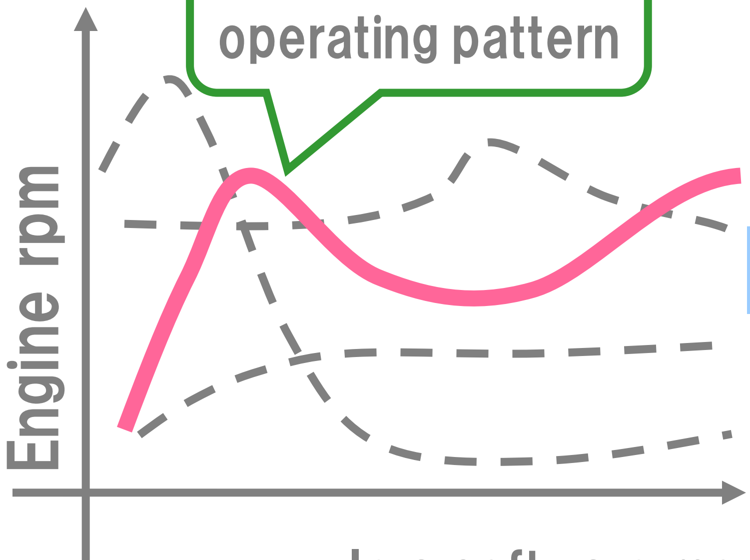
- ✓ Production
- ✓ Cyber Physical System
- ✓ Closed loop controller

Purpose: Establish prevention process with advanced V&V

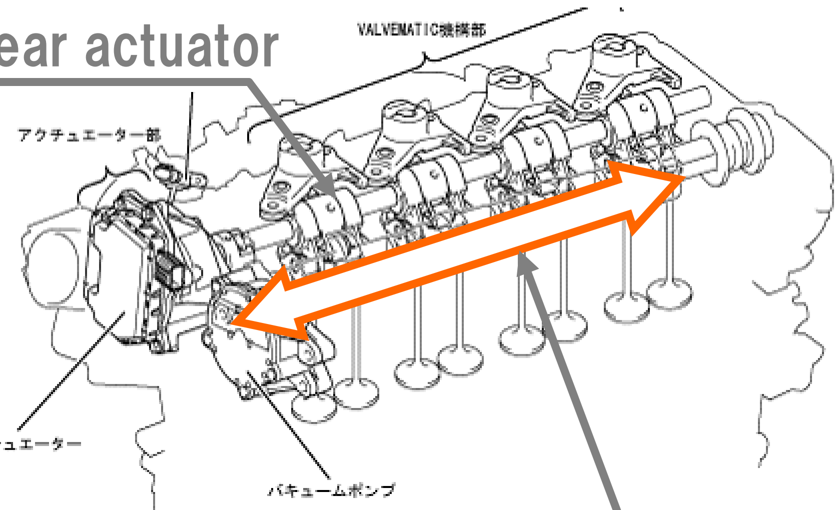
Applying model checking to our CPS

An issue occurred when we were developing.

Under a rare operating pattern



Linear actuator



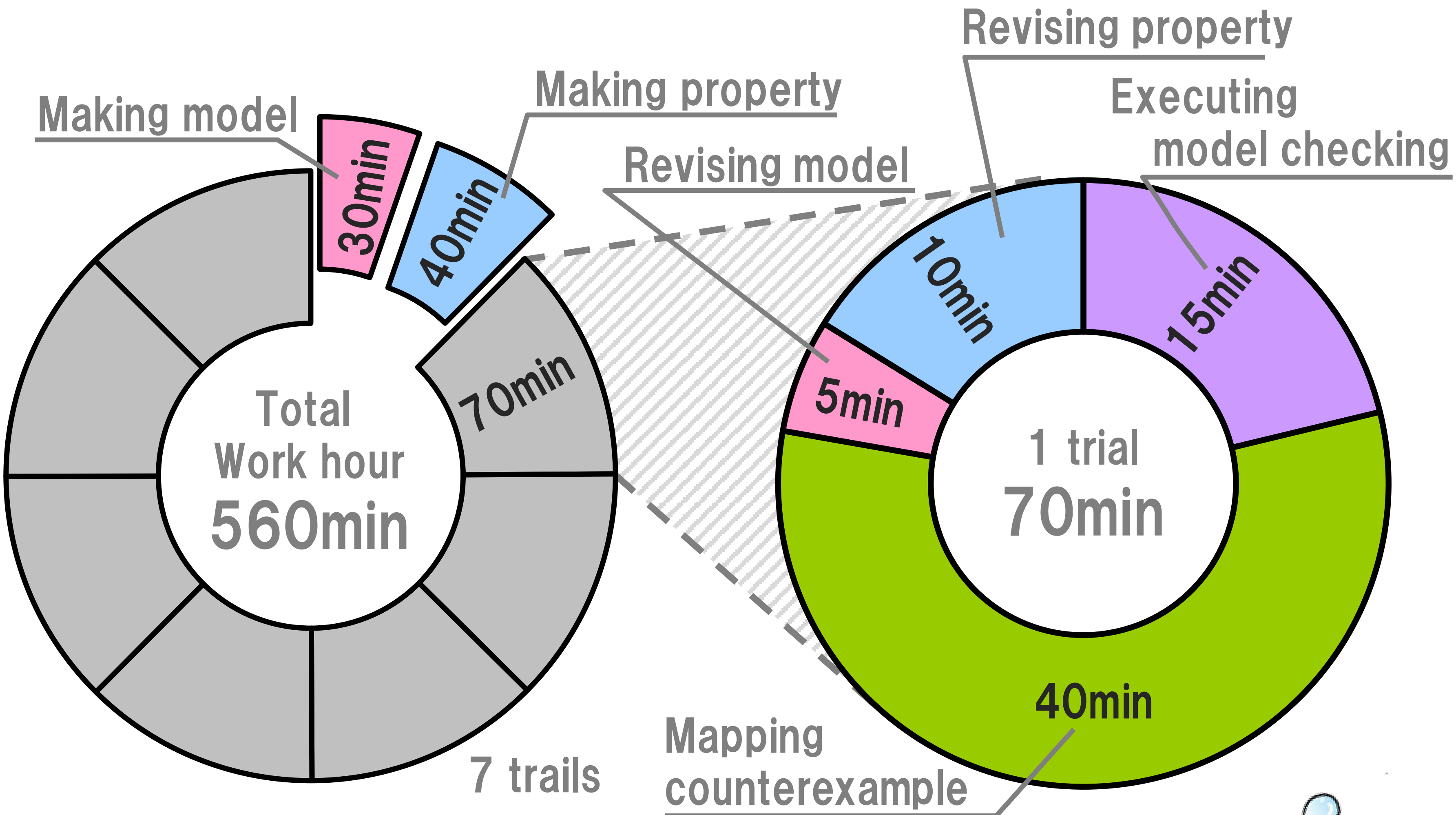
In a software module of linear sensor calibration

[Tomoya Yamaguchi,
Embedded System Symposium 2012,
(Japanese).]

An issue happened

I applied model checking to this issue and analyzed.

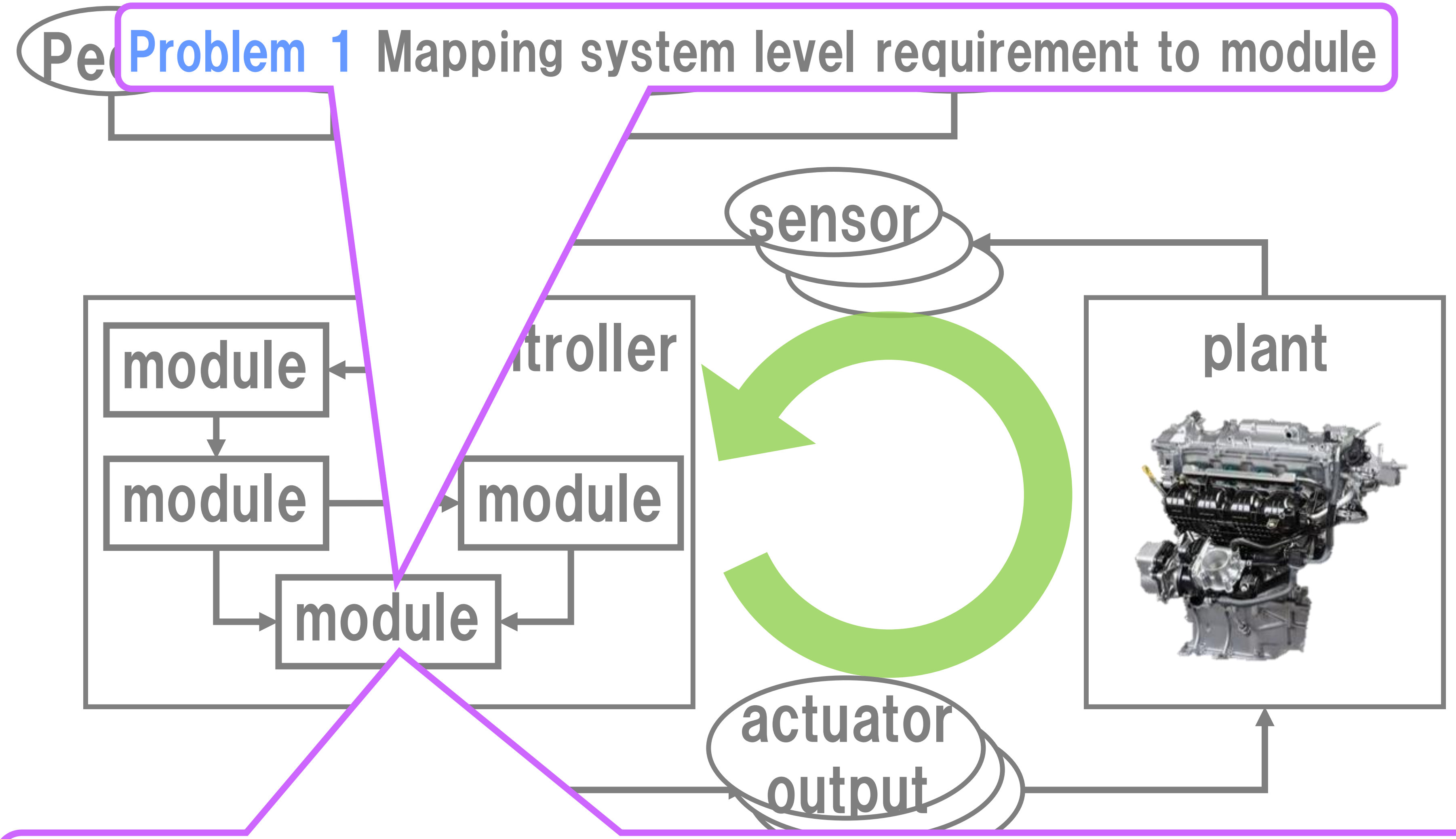
Applying model checking to our CPS



Making/revising property: 110 min
Mapping counterexample: 280 min for just 1 module

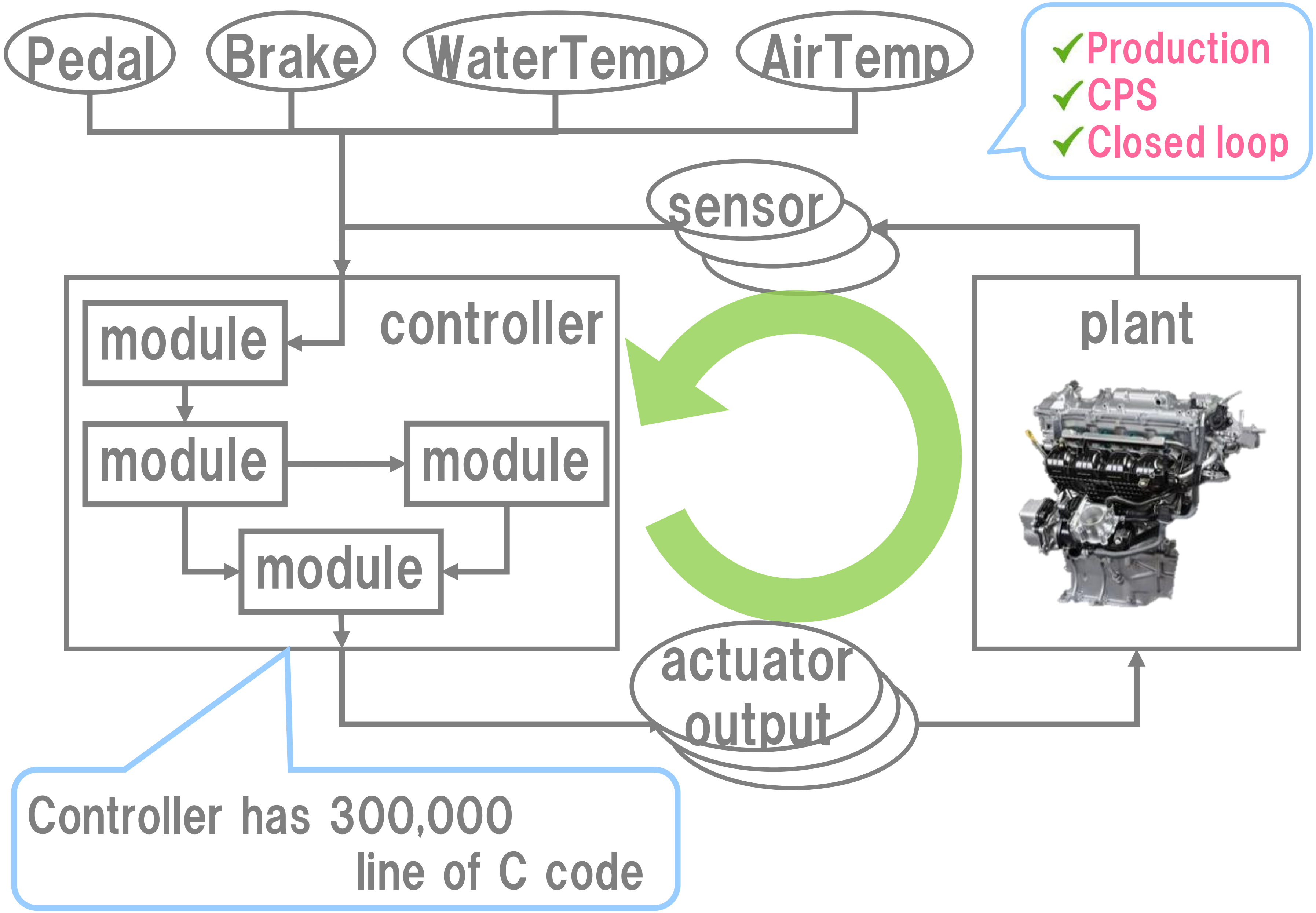
The problem of applying model checking

Problem 1 Mapping system level requirement to module

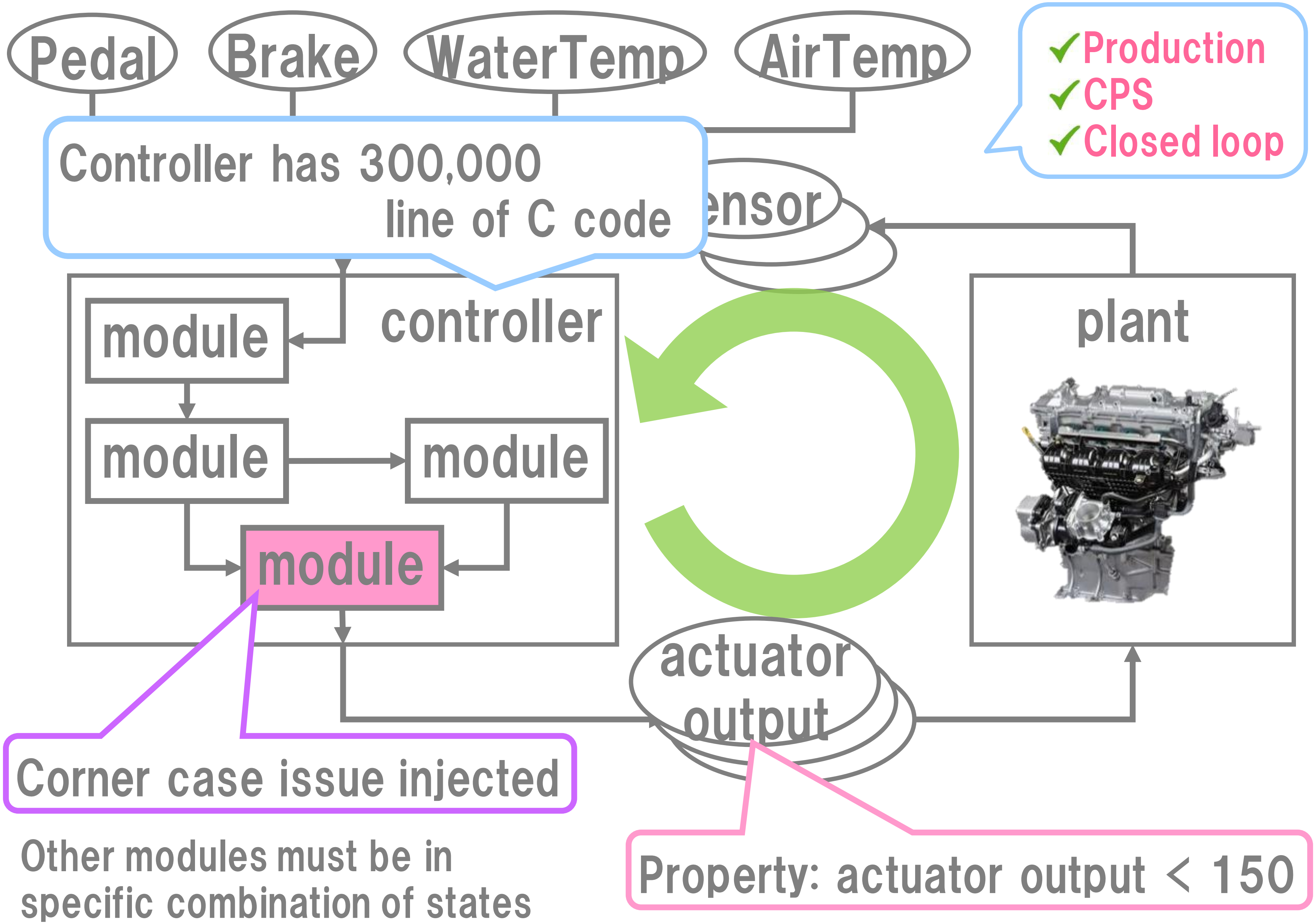


Problem 2 Mapping counterexamples found at the module level
to system-level counterexamples

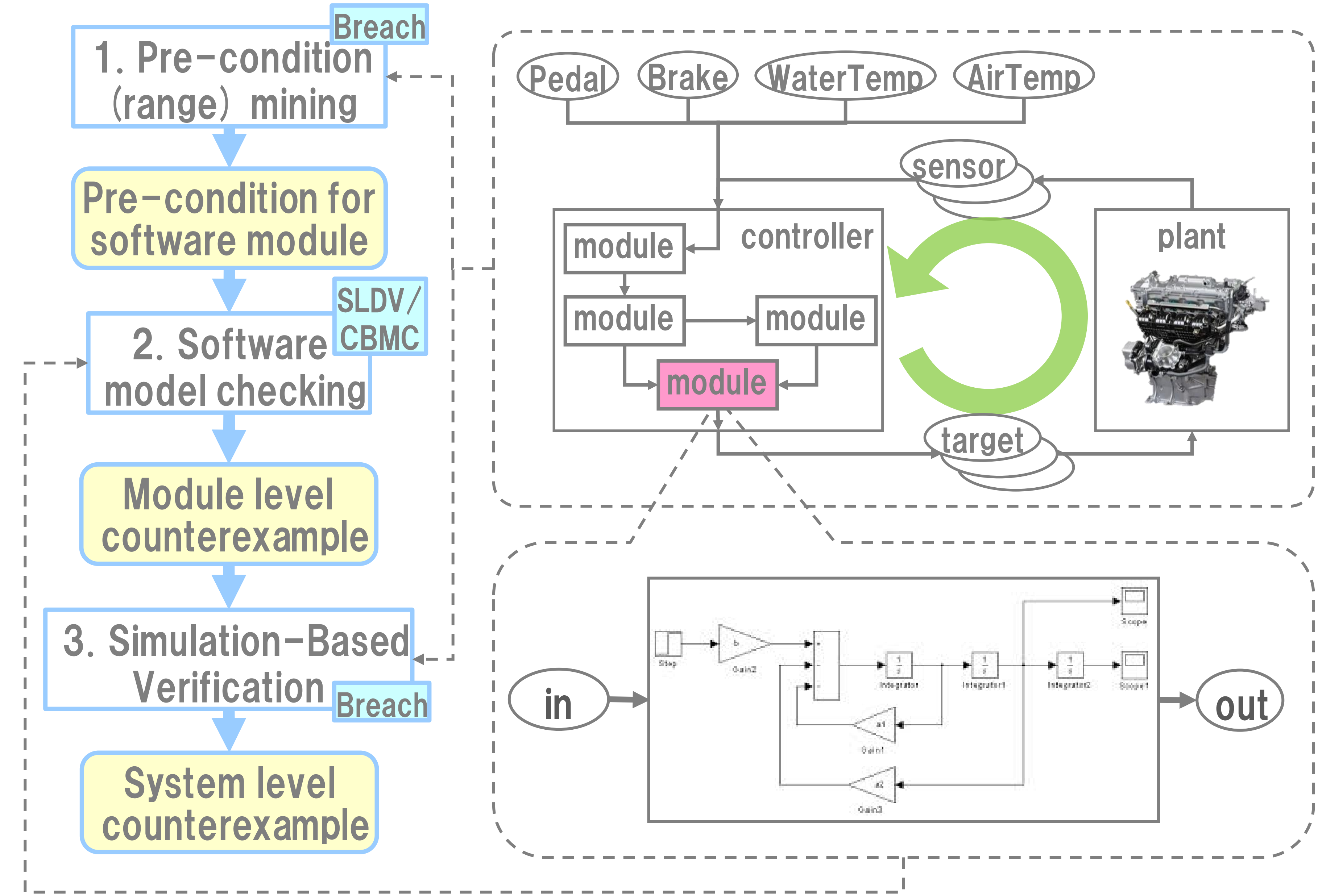
V&V object: Injected issue on actual Engine SILS



V&V object: Injected issue on actual Engine SILS

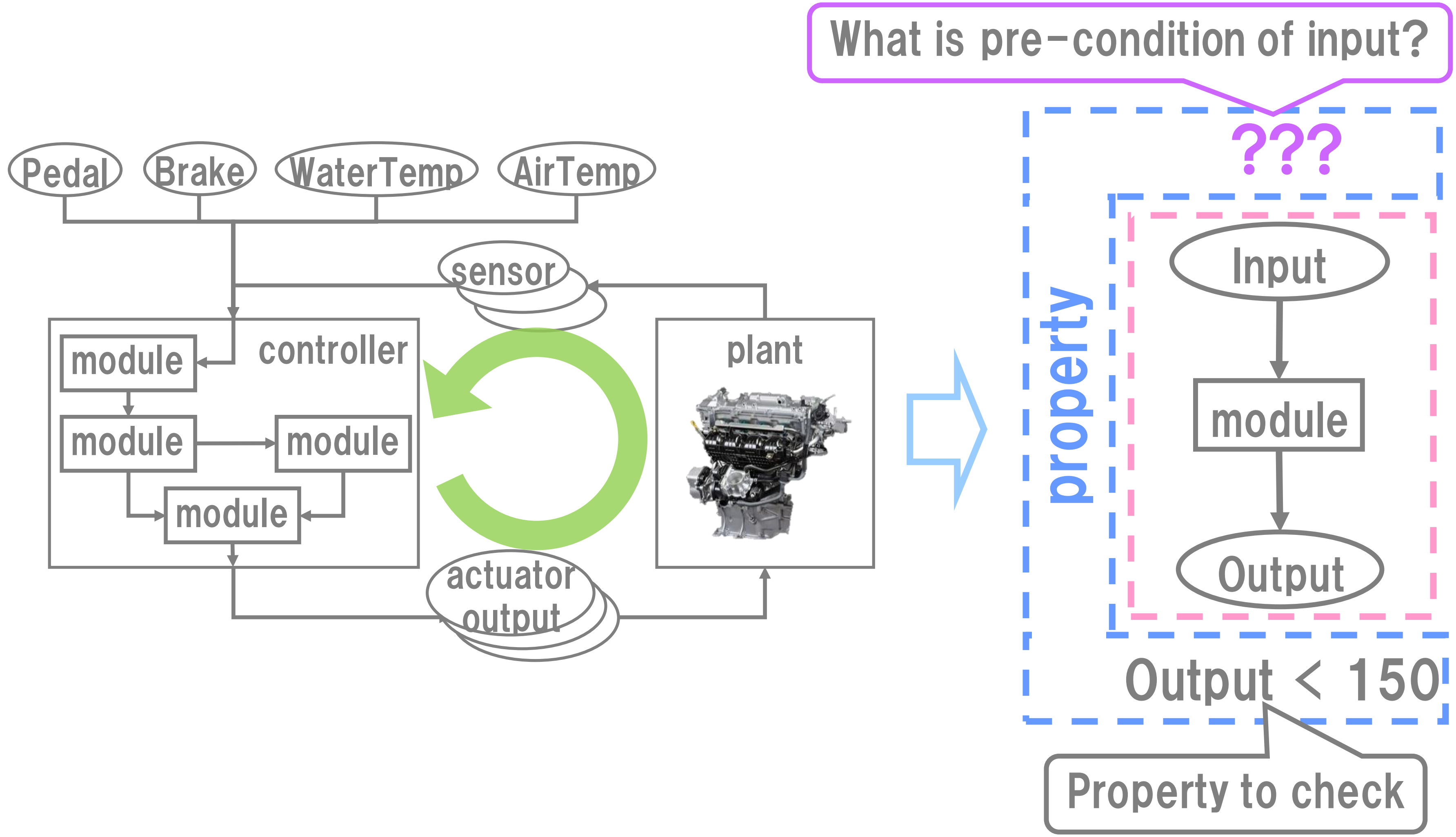


Overview of our methodology



Problem.1 Mapping system level requirement to module

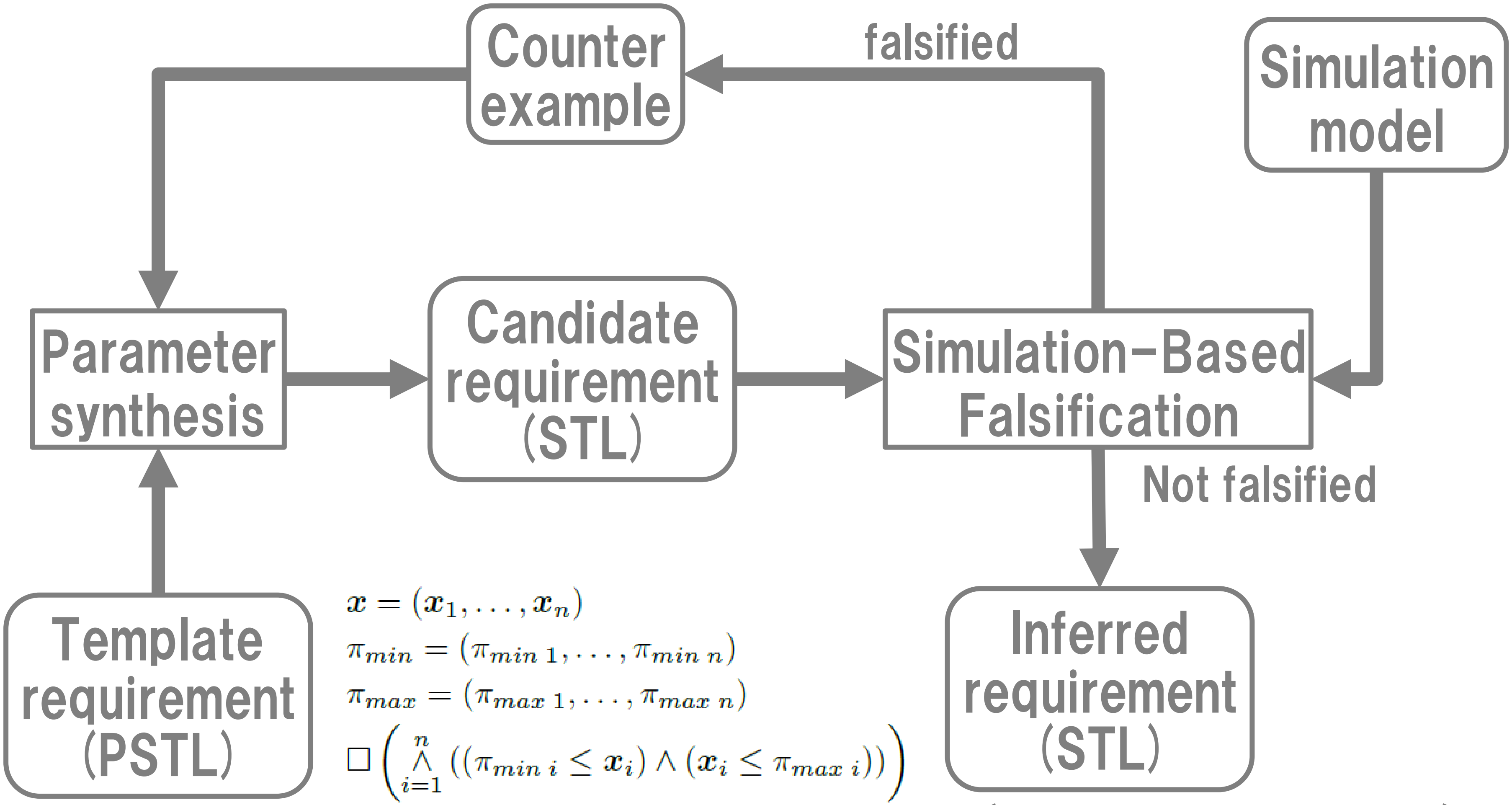
We have system level requirement



Hard to map system level requirement to module level

Counter measure for problem 1:Requirement Mining

[Xiaoqing Jin, et al., HSCC 2013]



$$\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$$
$$\pi_{min} = (\pi_{min\ 1}, \ldots, \pi_{min\ n})$$
$$\pi_{max} = (\pi_{max\ 1}, \ldots, \pi_{max\ n})$$
$$\Box \left(\bigwedge_{i=1}^{n} ((\pi_{min\ i} \le \boldsymbol{x}_i) \wedge (\boldsymbol{x}_i \le \pi_{max\ i})) \right)$$

$\Box((-100 < x_1) \wedge (x_1 < 100) \ldots)$

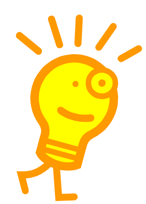
Apply requirement mining to mine pre-condition (range)

Result of using module level requirement

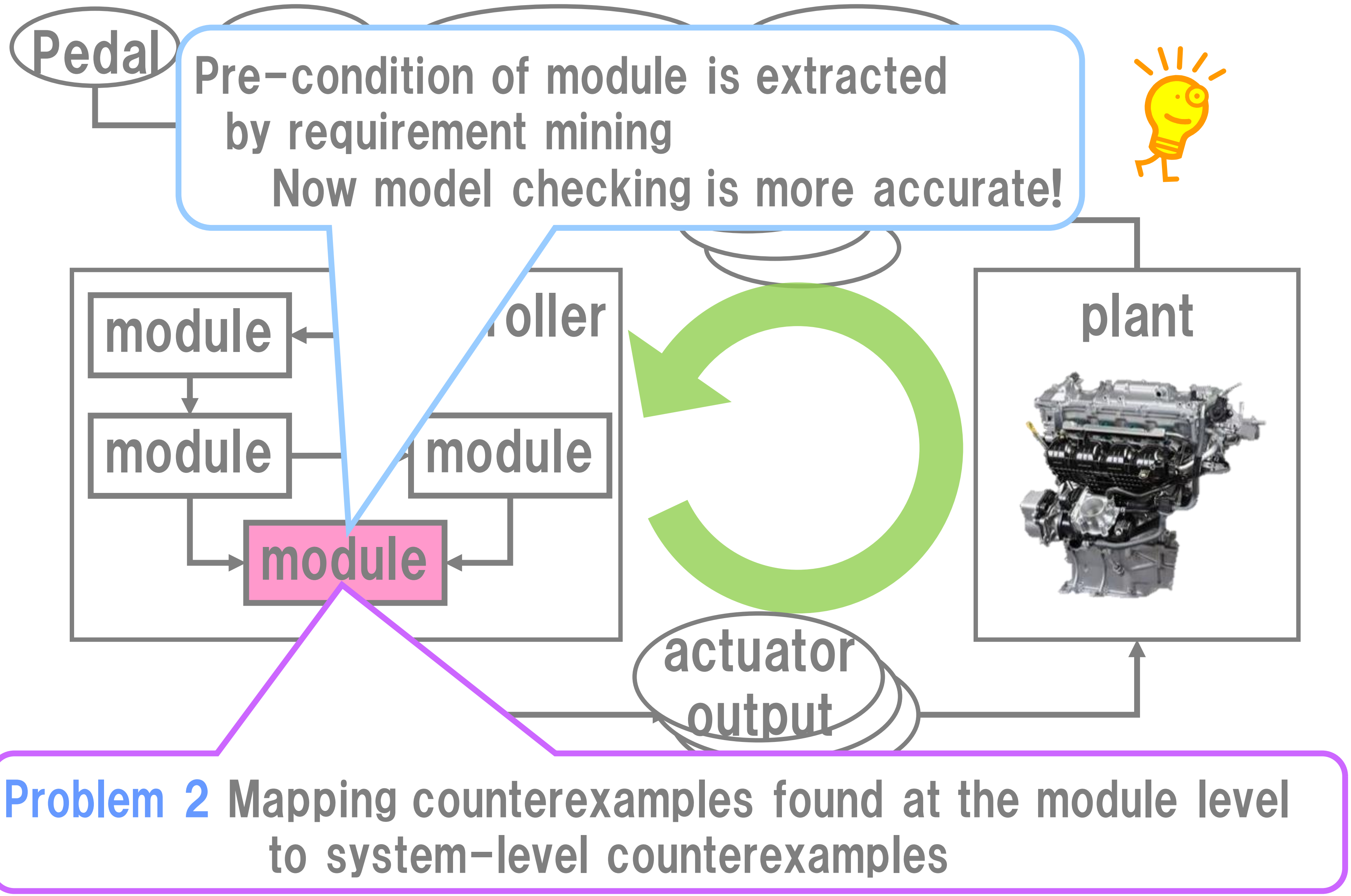
Counterexample comes from model checking

Input variable	No range	With range mining	
	counterexample	range	counterexample
waterTemp [℃]	89.4	[-30.0, 100.0]	90.0
atmosphericPressure [bar]	3.5	[0.0, 1.0]	1.0
gear	5	[0, 6]	6
gearHoldFlag	0	0	0
idlFlag	0	[0, 1]	0
catalystTempHIGHflag	1	[0, 1]	1
fuelCutFlag	0	[0, 1]	0
engRpm [rpm]	2600.0	[0.0, 5310.9]	2600.0

false positive case is avoided by using range mining

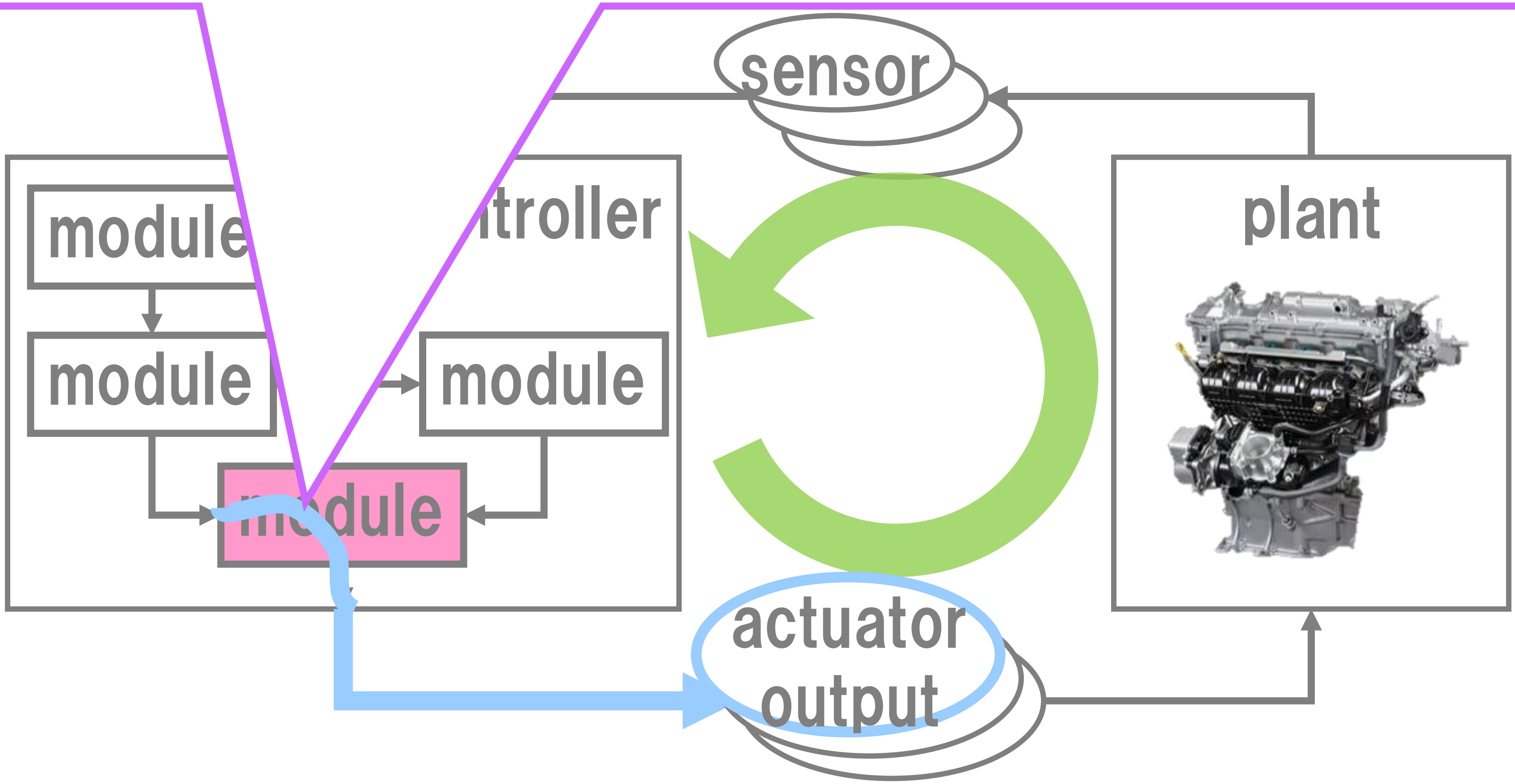


Problem 2 Mapping counterexamples found at the module level to system-level counterexamples



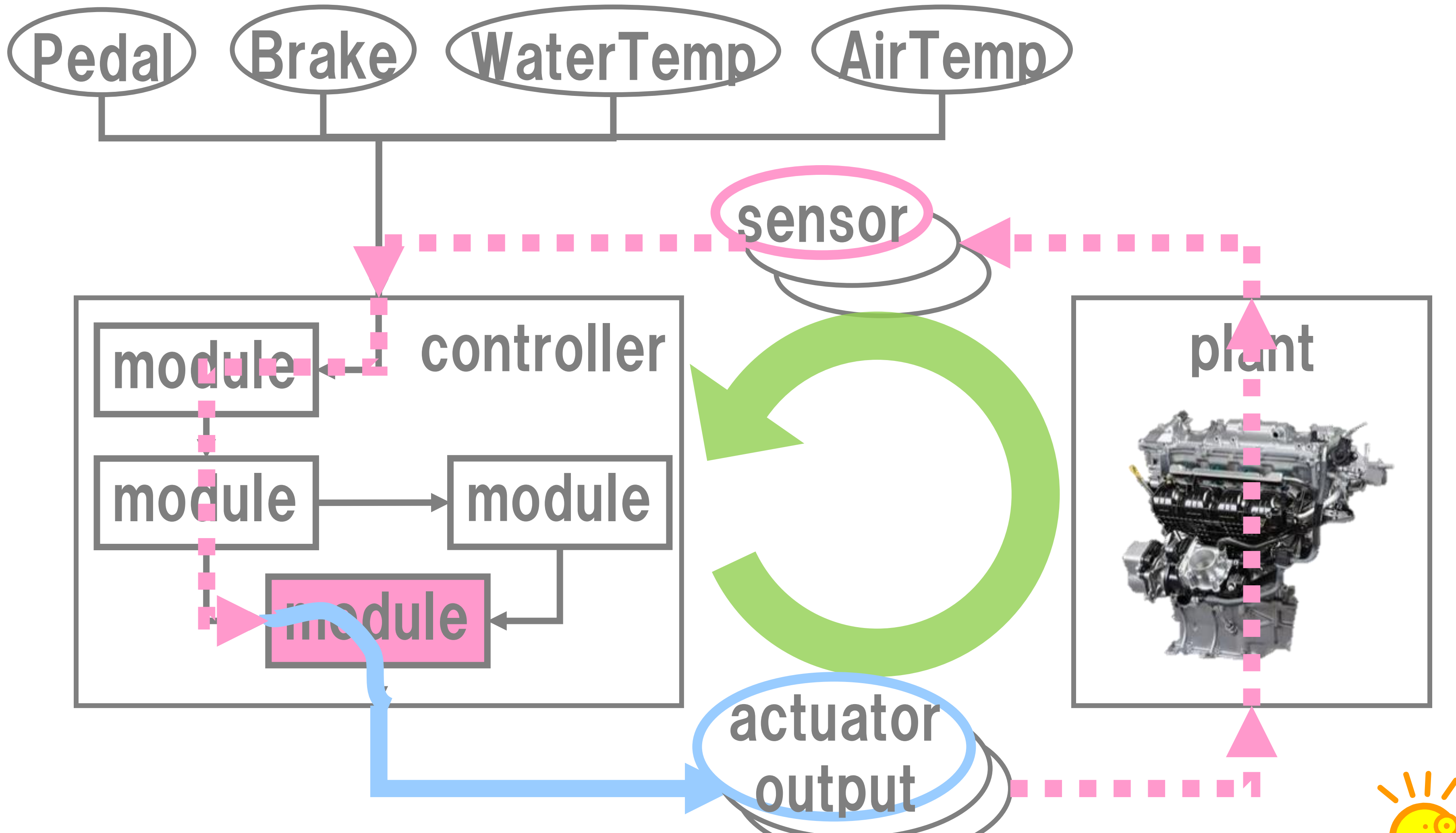
Problem 2 Mapping counterexamples found at the module level to system-level counterexamples

Is this module level counterexample from model checking false positive or true positive?



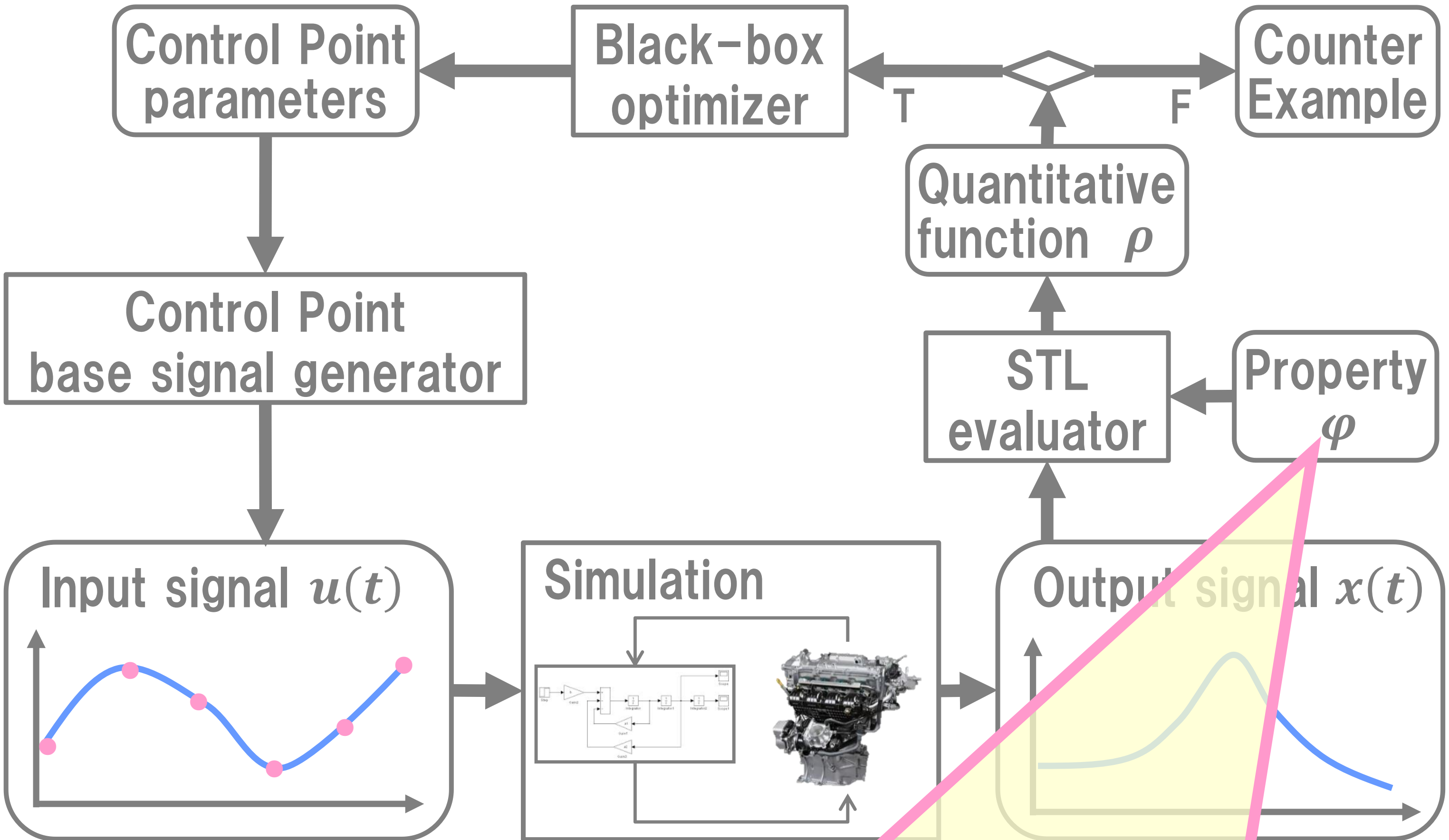
Generally, it needs much work-hour, HI-level V&V skill and system knowledge

Problem 2 Mapping counterexamples found at the module level to system-level counterexamples



Hypothesis: Module level CE is a true positive, when system level CE containing module level CE is found

Simulation-Based Verification with cost function

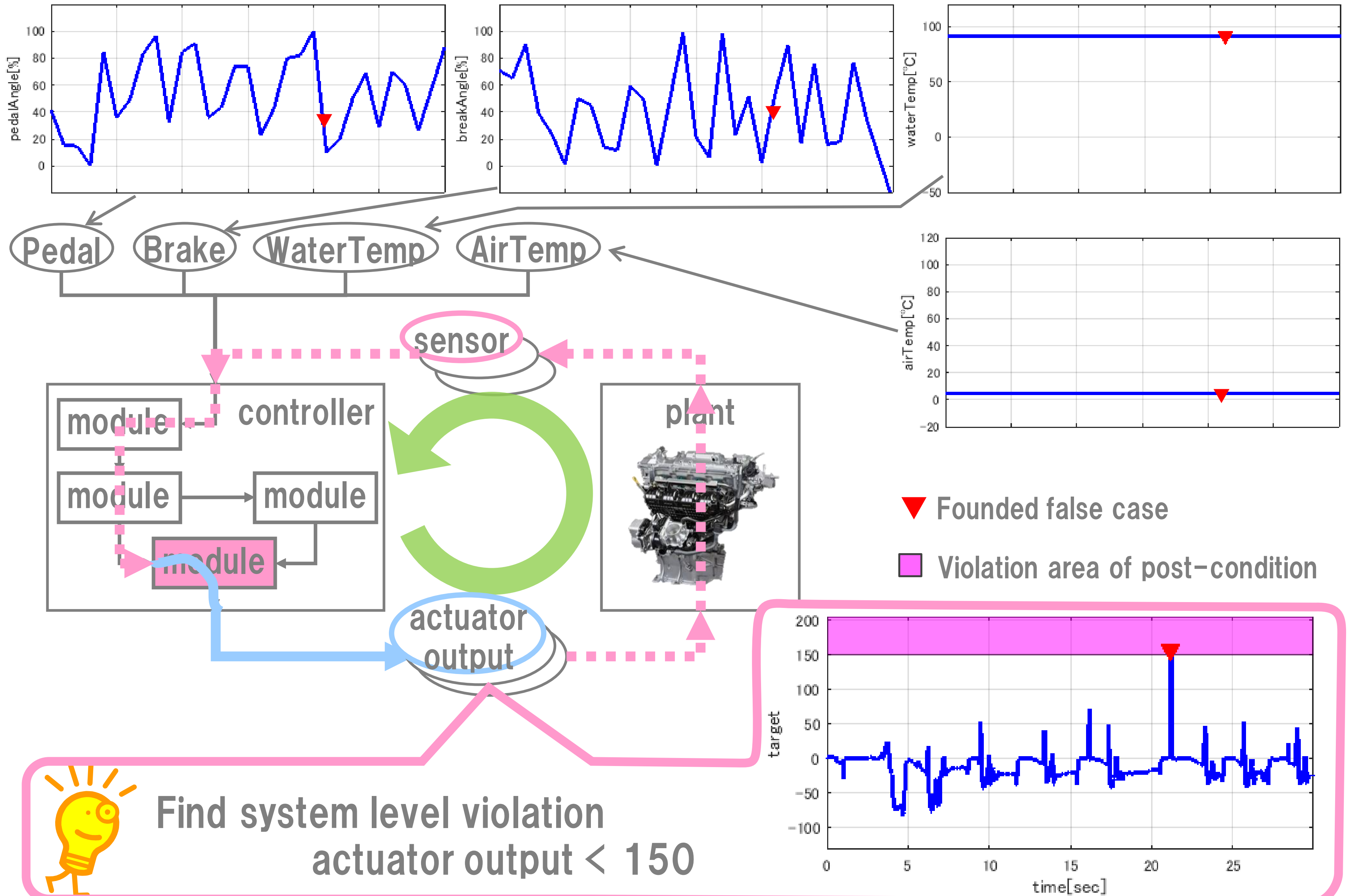


Drive system to module level CE using Simulation-Based Verification

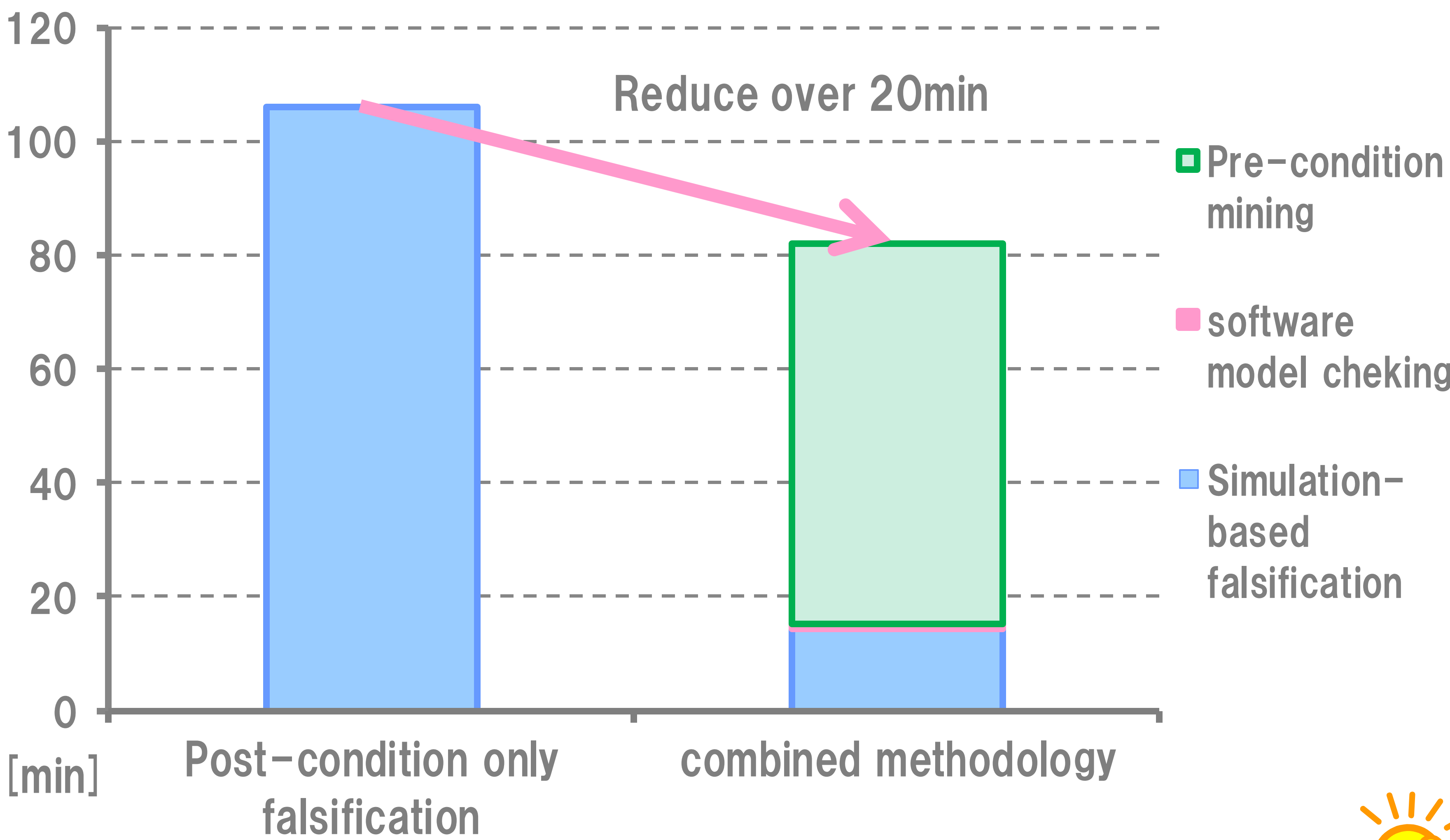
Want to falsify property:
(minimize distance to CE)

$$\varphi(x) = \Box\left(\sqrt{\sum_{i=1}^{n}(x_i(t) - \hat{x}_i)^2} \geq \varepsilon\right)$$

Found system level corner case issue



Comparison with just Simulation-based Falsification



significantly more effective than using just
software Model checking or just Search-based falsification

Conclusion

- We propose combined methodology
  (= Requirement Mining + Model Checking
  + Simulation-based verification)
- New methodology is applied to production closed loop CPS
- Our combined methodology can be significantly more effective than using just software Model checking or just Simulation-based verification

Special thanks

Breach: Breach is provided by U.C. Berkeley, Prof. Sanjit Seshia and Dr. Alexandre Donzé. Breach has flexible extendibility for the requirement mining and the simulated-base verification.

CBMC: CBMC is provided by Univ. Oxford, Prof. Daniel Kroening and Dr. Martin Brain. CBMC is a sophisticated tool and was greatly helpful for our case study.

SMiL: Toyota in-house engine SILS. Fujitsu-ten provides and also supports us well.