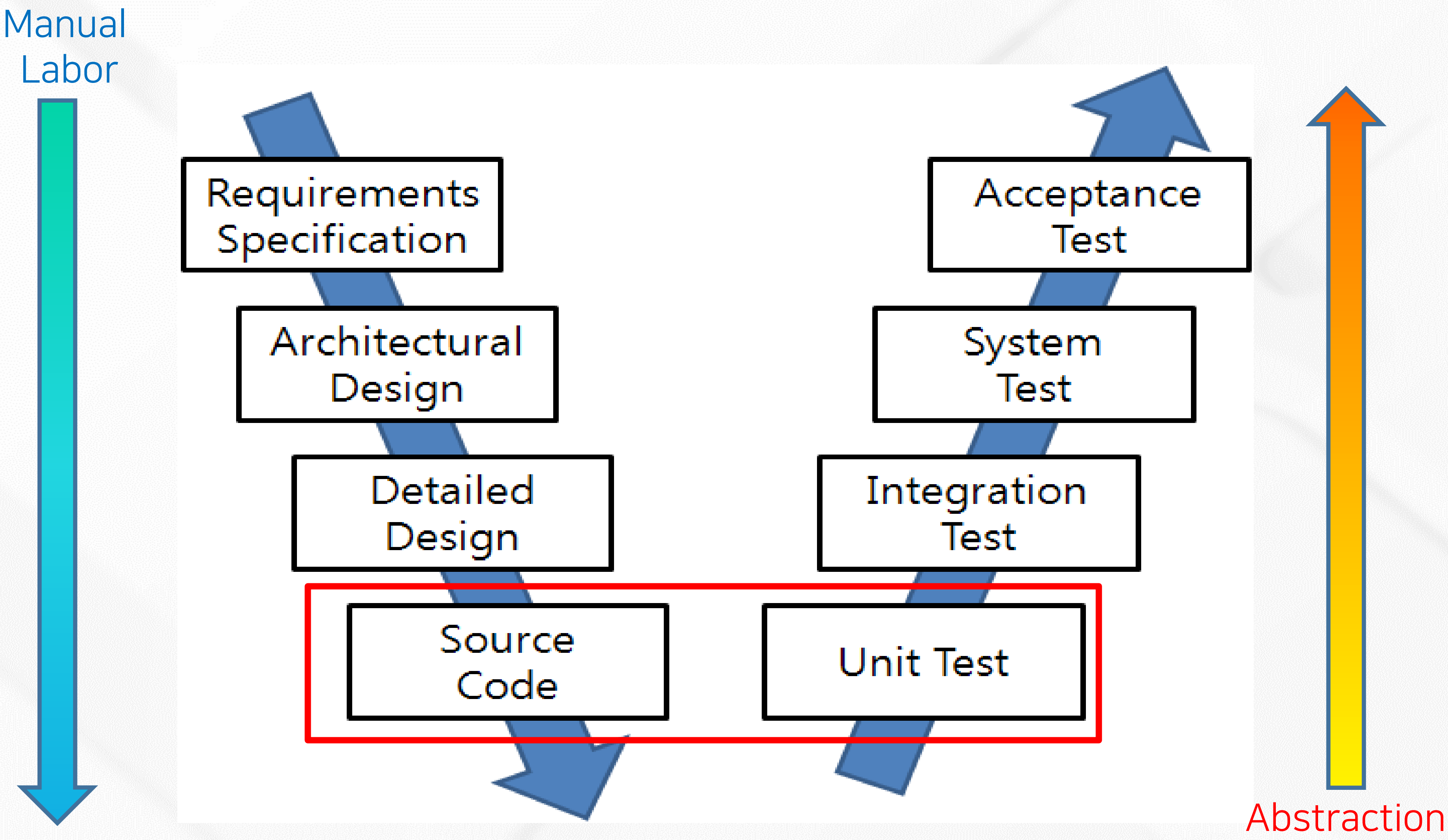
Automated Unit Testing

Moonzoo Kim
moonzoo.kim@gmail.com
KAIST

SW Development and Testing Model (V model)

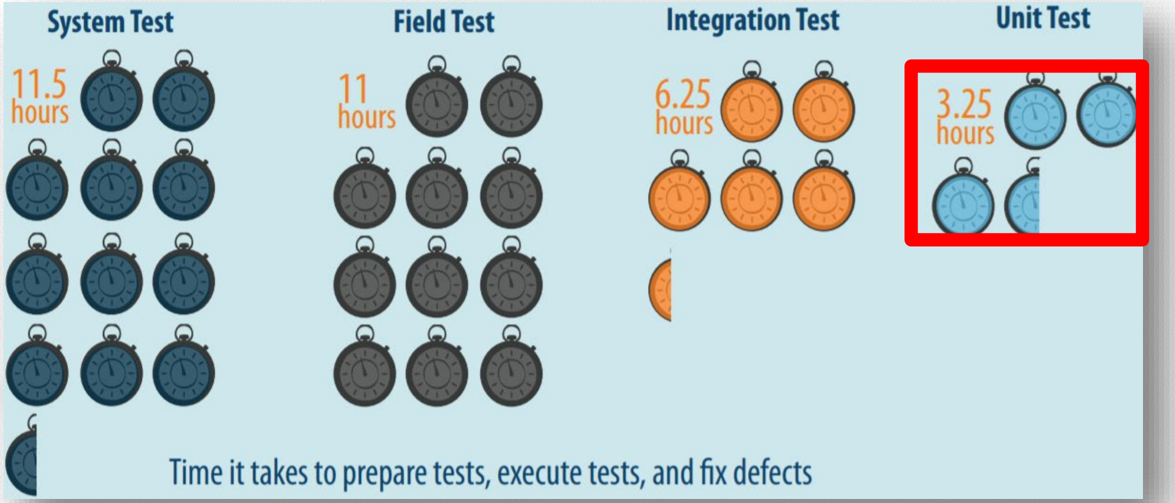


Many Benefits of Unit Testing

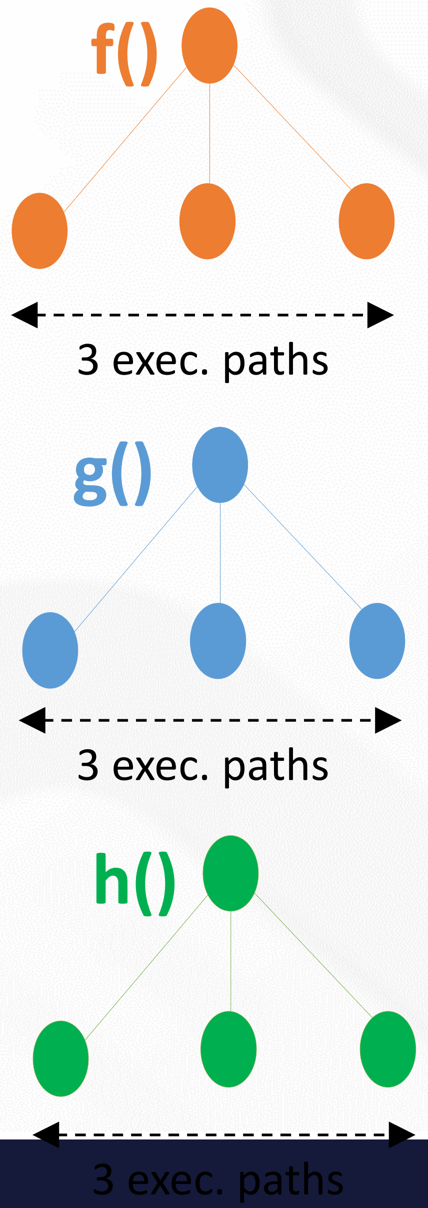
› **Bug correction cost: 7x cheaper** than system tests
  › $937 (unit test) vs $7,136 (system test)

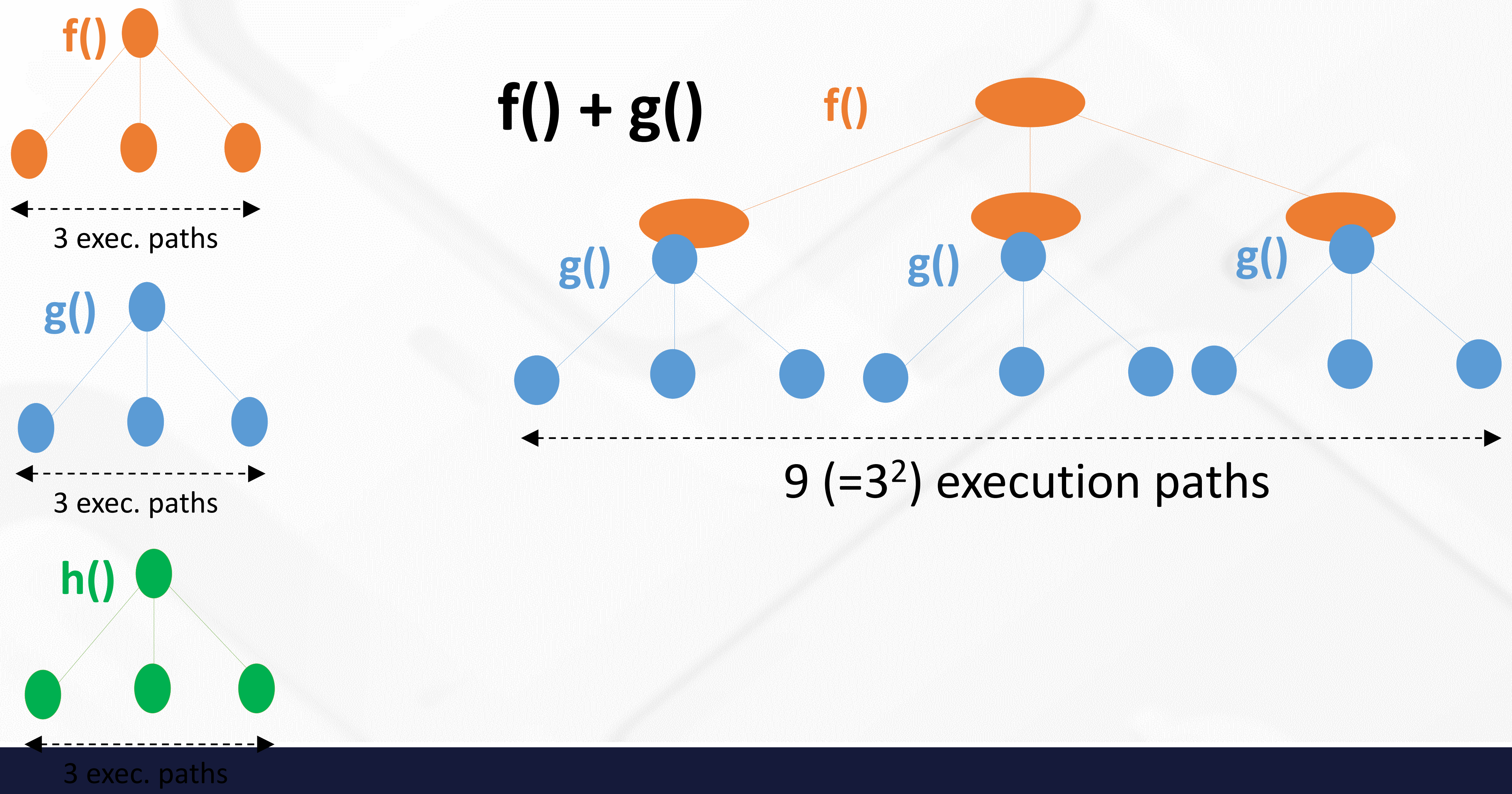
Source: B. Boehm and V. Basil, Software Defect Reduction Top 10 List, IEEE Computer, January 2001

› **Bug correction time: 3x faster** than system testing
  › 3.25 hours vs 11.5 hours

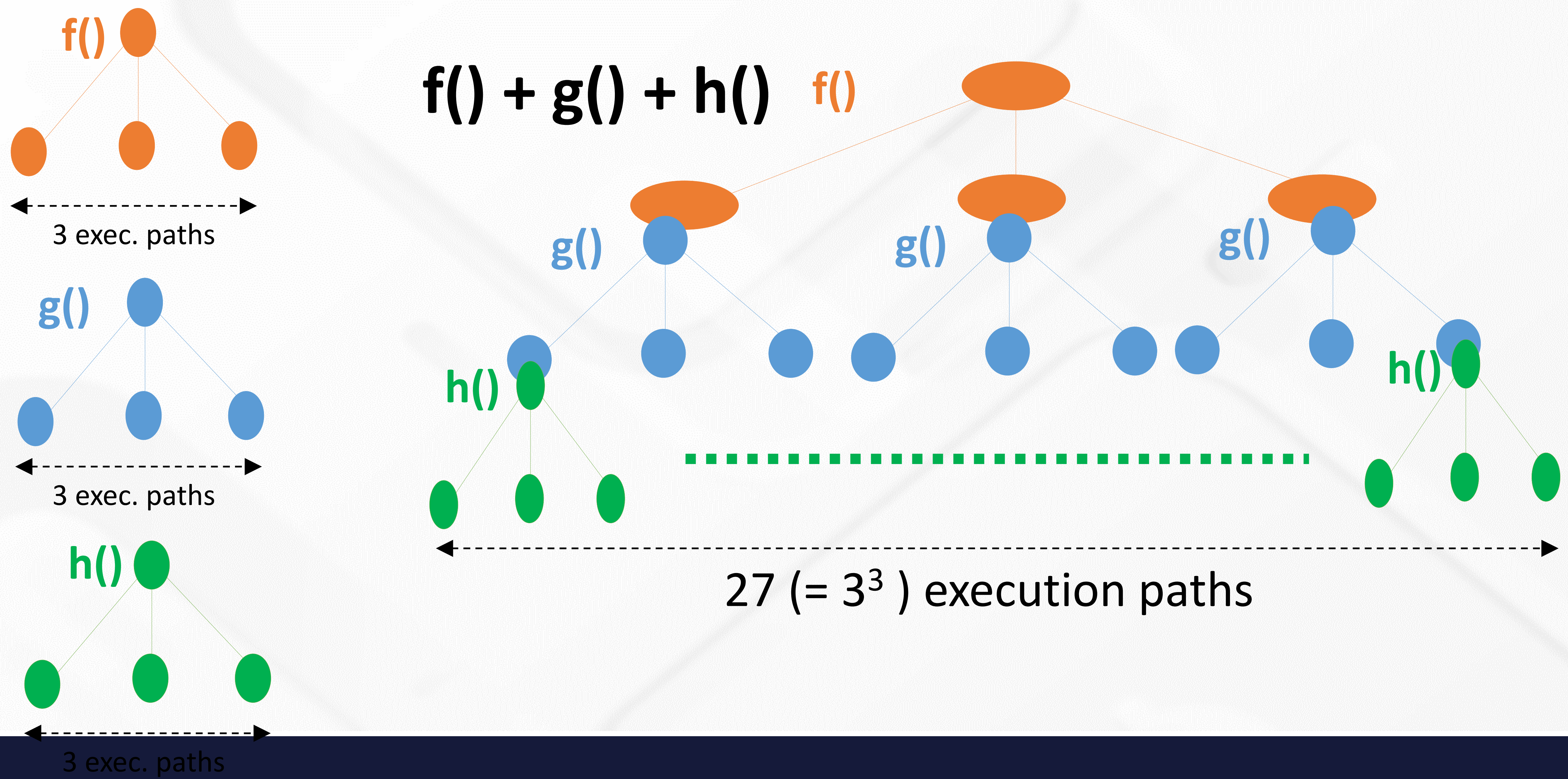


Source: Capers Jones, Applied Software Measurement: Global Analysis of Productivity and Quality


f()
3 exec. paths
g()
3 exec. paths
h()
3 exec. paths

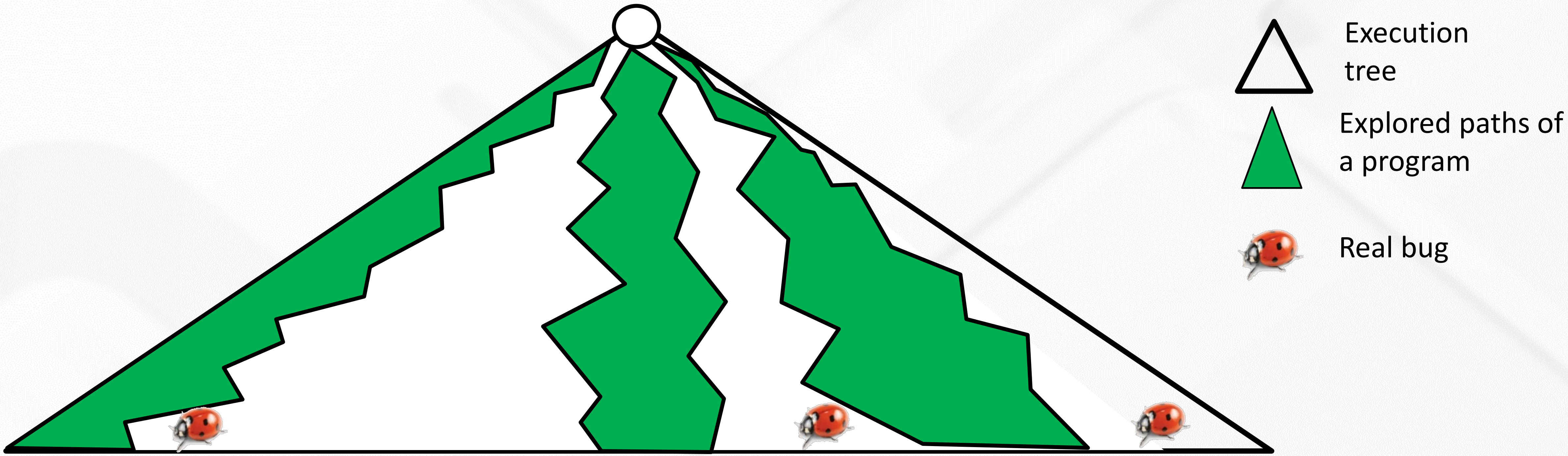


f()
3 exec. paths
g()
3 exec. paths
h()
3 exec. paths
f() + g()
f()
g()
g()
g()
9 (=3²) execution paths


f()
3 exec. paths
g()
3 exec. paths
h()
3 exec. paths
f() + g() + h()
f()
g()
g()
g()
h()
h()
27 (= 3³) execution paths

Pros and Cons of Auto. Test Gen. in **System-level**

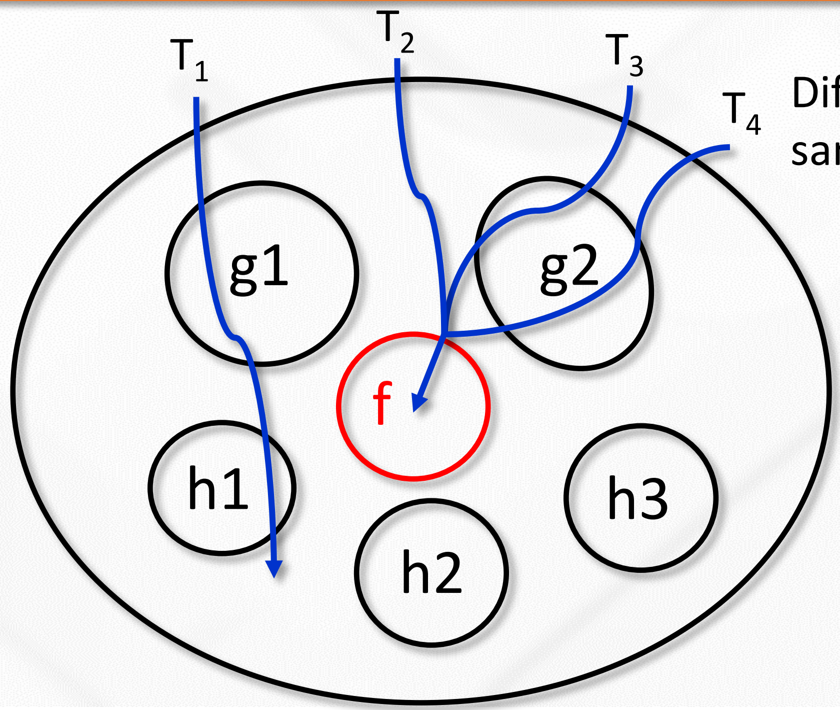
- Pros: **No false alarms**
- Cons: Low bug detection power due to **large search space**



Automated Test Generation in System-Level

Pros
+ Can be easy to generate system TCs due to clear interface specification
+ No false alarm (i.e., no assert violation caused by infeasible execution scenario)

Cons
- Low controllability of each unit
- Large and complex search space to explore in a limited time
- Hard to detect bugs in corner cases

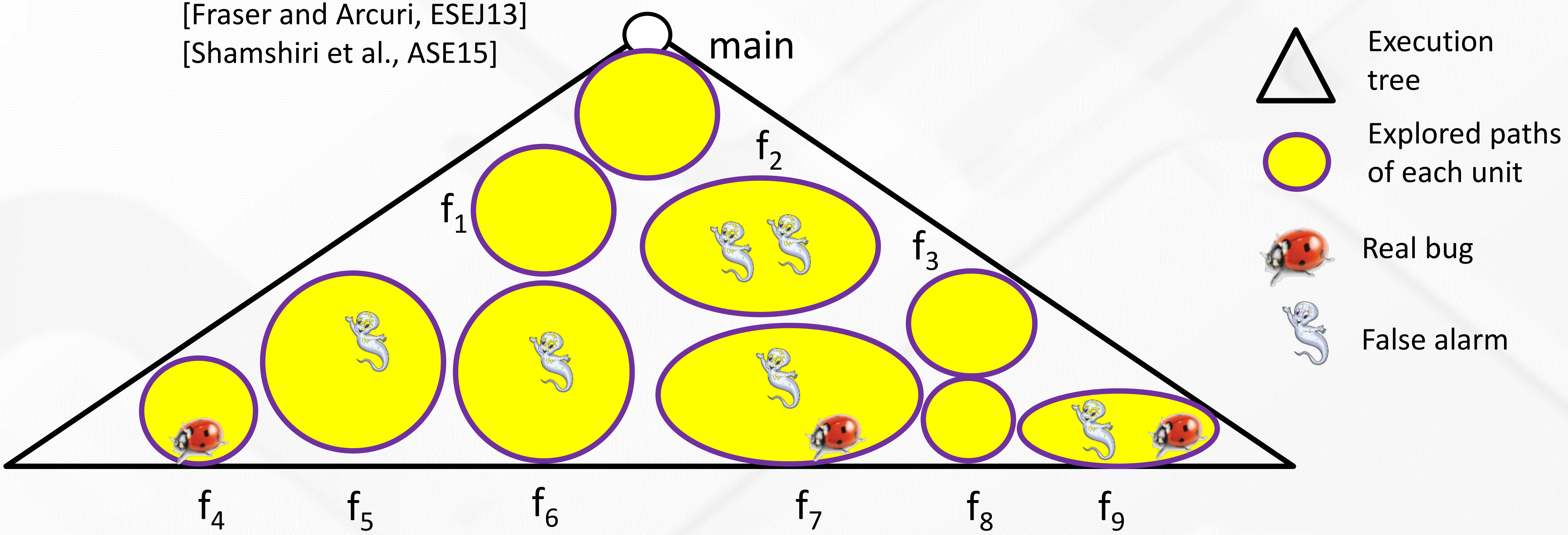


Different system tests T_2 to T_4 exercise the same behavior of the target unit

Pros and Cons of Auto. Test Gen. in **Unit-level**

- Pros: High bug detection power for **small search space**
- Cons: **Many false alarms** due to over-approximated context of a unit

[Gross et al., ISSTA12]
[Fraser and Arcuri, ESEJ13]
[Shamshiri et al., ASE15]



Automated Test Generation in Unit-Level

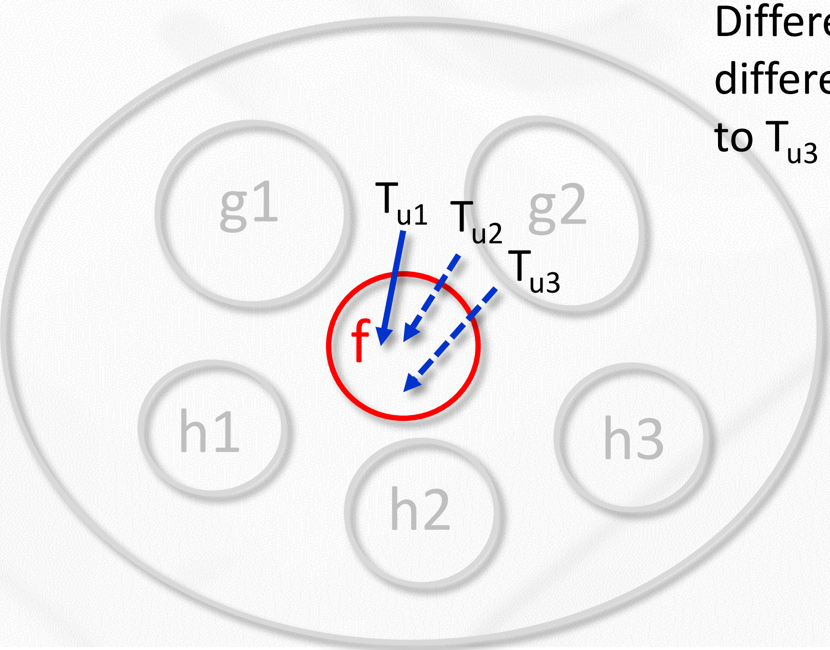
Pros

+ High controllability of a target unit
+ Smaller search space to explore than system testing
+ High effectiveness for detecting corner cases bugs

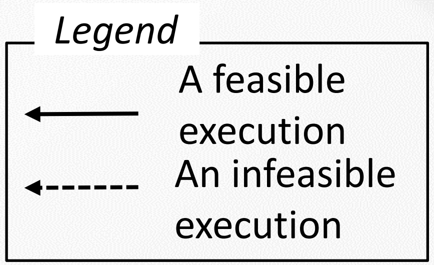
Cons

- Hard to write down accurate unit test drivers/stubs due to unclear unit specification

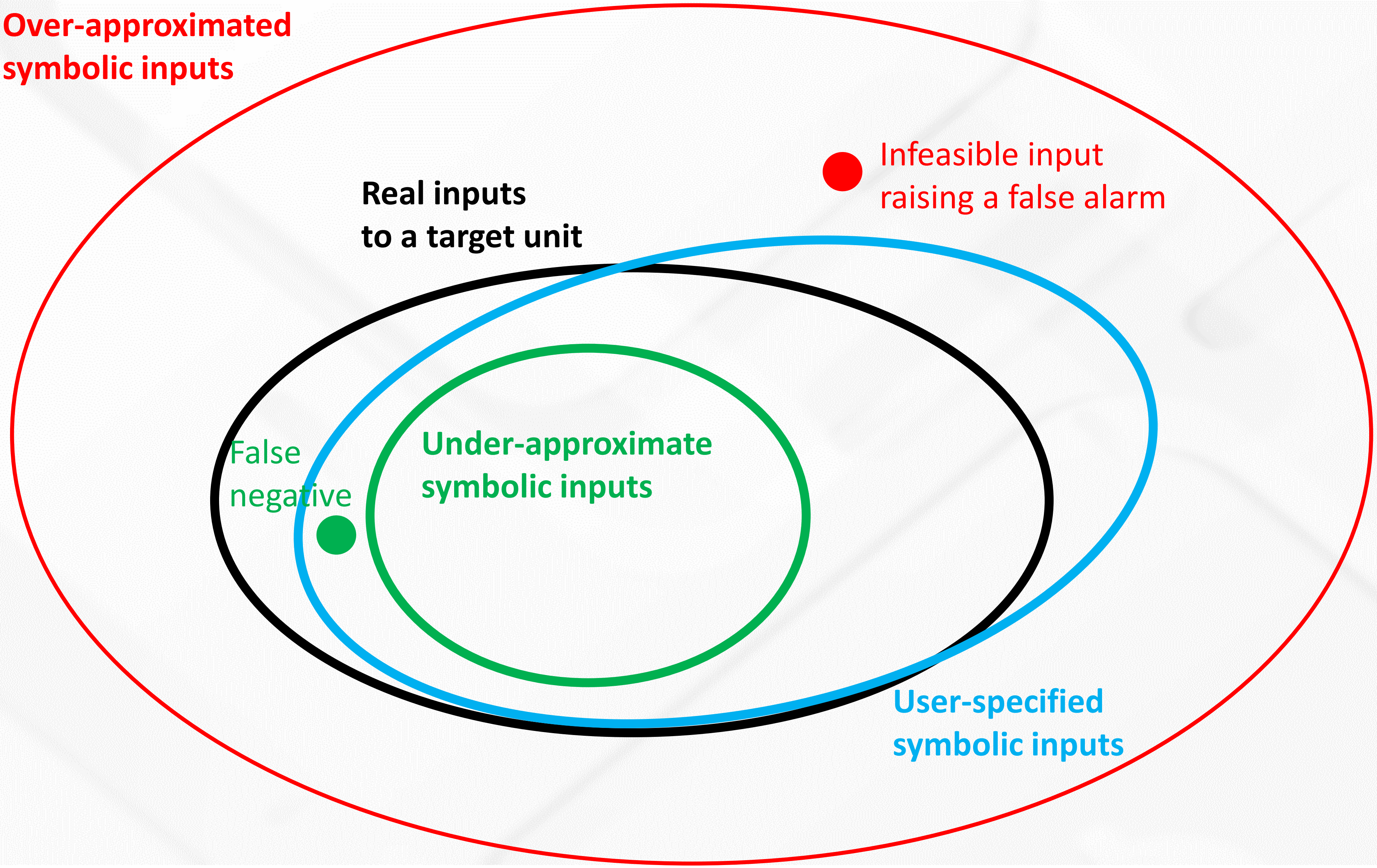
- High false/true alarm ratio



Different unit tests T_{u1} to T_{u3} directly exercise different behaviors of the target unit, but T_{u2} to T_{u3} exercise infeasible paths



Approximate Input Space for Symbolic Unit Testing



Related Work on Automated Unit Testing

	Bug detection ability	False/True alarm ratio	Target languages
Function input generation [PLDI 05][FSE 05][EMSOFT 06][TAP 08][ISSTA 08][SEC 15]	High	High	Procedural or OO languages
Method-sequence generation [ICSE 07] [ICST 10][FSE 11] [ICSE 13]	High	Medium	Object-oriented languages
Capture system tests to generate unit tests [TSE 09] [STTT 09][ISSTA 10]	Low	Low	Object-oriented languages
Automated Unit Test Generation with Realistic Unit Context	High (86.7% of target bugs in SIR and SPEC2006)	Low (2.4 false alarms per one true alarm)	Procedural languages

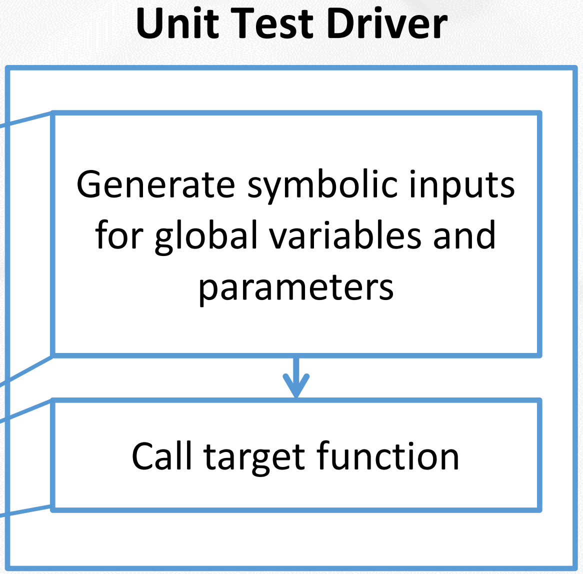
Constructing Unit Test Driver/Stubs

› Example of an automatically generated unit-test driver

```
typedef struct Node_{
  char c;
  struct Node_ *next;
} Node;
Node *head;
// Target unit-under-test
void add_last(char v){
  // add a new node containing v
  // to the end of the linked list
  ...}
// Test driver for the target unit
void test_add_last(){
  char v1;
  head = malloc(sizeof(Node));
  SYM_char(head→c);
  head→next = NULL;
  SYM_char(v1);
  add_last(v1); }
```

Set global variables

Set parameter



Trade-off between Bug Detection Ability and Accuracy

Bug Detection **Ability**
(aiming low false negative)

Recall

Bug Detection **Accuracy**
(aiming low false positive)

Precision

Cutting-edge Accuracy of Unit Testing

Randoop: 5.9:1
[Gross et al., 2012]
Evosuite: 6.3:1
[Fraser and Arcuri, 2013]
UC-KLEE: 5.7:1
[Ramos and Engler, 2015]

OOP features

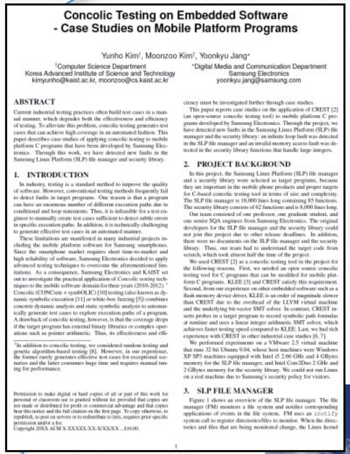
Manual Assumption

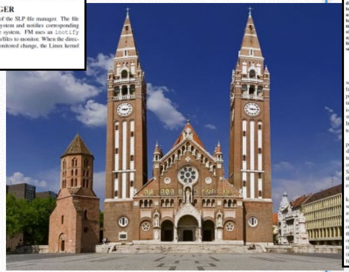
CONBRIO [Kim'18]:
4.5:1 w/ 91% of bugs detected

Concolic Unit Testing Project w/ Samsung DMC for 5 years

- Goal: To detect bugs in Samsung smartphones
- Project period: '10~'14
- Project funding: 400,000 USD
- Results:
- Developed Concolic unit-testing tool **CONBOL**
- Detected **many crashes** in 4 MLOC smartphone SW

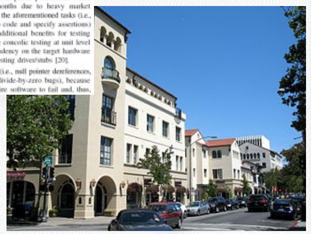
Industrial project progress: 2010 → 2011 → 2012 → 2013 → 2014





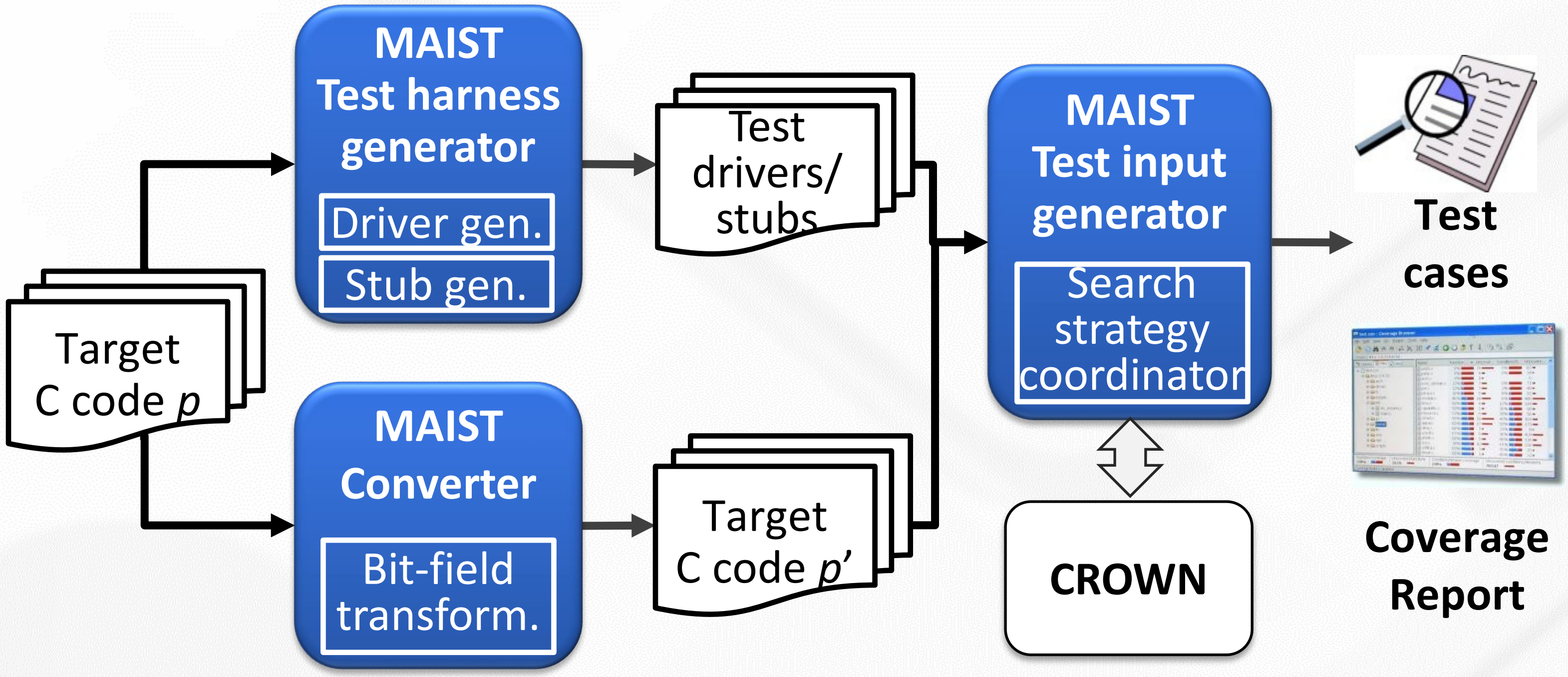
- FSE '2011

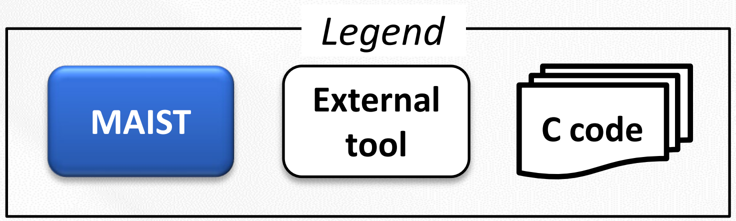
- ICSE 2012, ICST 2012



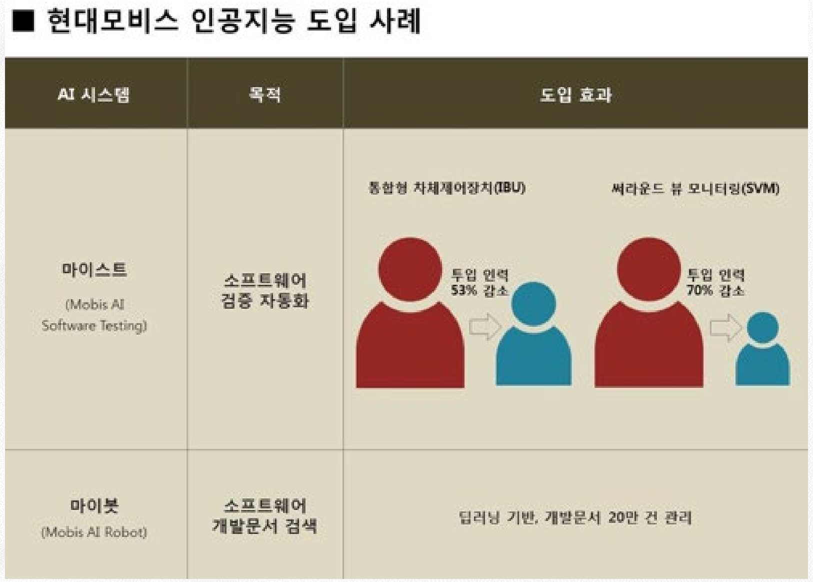
- ASE 2013

Concolic Unit Testing Project w/ Hyundai and Mobis for 5 years





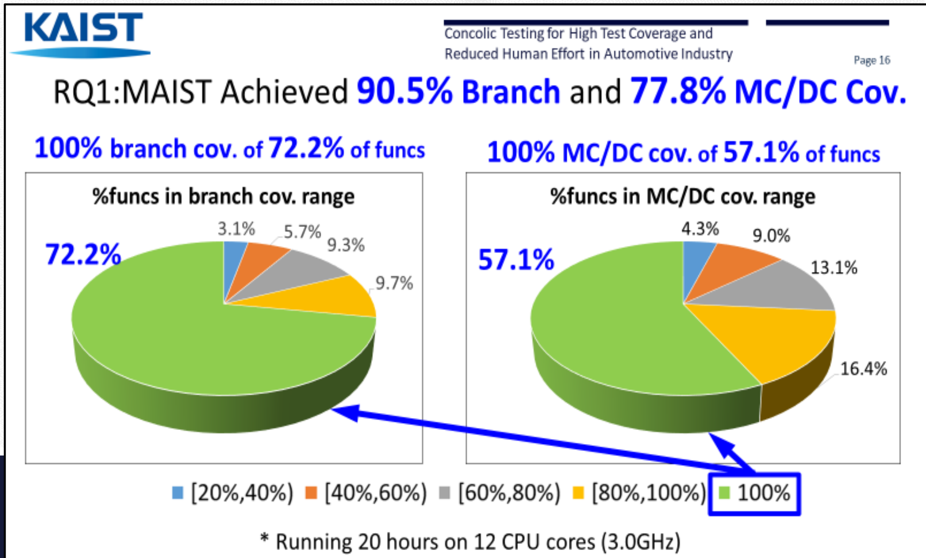
CROWN features
- Bit-level accuracy
- Easy-to-extend
- Low memory overhead



■ 현대모비스 인공지능 도입 사례

AI 시스템	목적	도입 효과
마이스트 (Mobis AI Software Testing)	소프트웨어 검증 자동화	통합형 차체제어장치(IBU) 투입 인력 53% 감소; 써라운드 뷰 모니터링(SVM) 투입 인력 70% 감소
마이봇 (Mobis AI Robot)	소프트웨어 개발문서 검색	딥러닝 기반, 개발문서 20만 건 관리

http://m.yna.co.kr/kr/contents/?cid=AKR20180720158800003&mobile



Practical Benefit of Automated Unit Testing

System testing is **expensive** and **less effective**

- **Full vehicle HW** and **human drivers** are required
- Driving a car with **various physical environments** spends **a lot of time**
- **Hard-to-achieve high test coverage** due to low controllability

VS

Unit testing is **cheap** and **more effective**

- Full vehicle HW and human drivers are **NOT necessary**
- No need to drive a car
- **Easy-to-achieve high coverage** due to high controllability

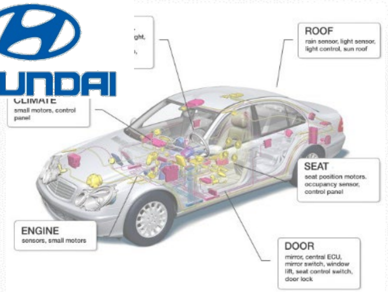
Solution for Huge Economic & Social Cost **due to SW Bugs**

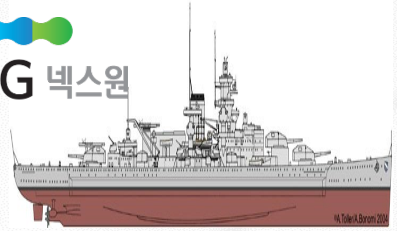
Labor-intensive Manual Testing
Large SW Testing Cost and Time
Low Bug Detection Abilty
Low Product Quality

Solution: AI-based Automated Concolic Testing Technique

movie link https://bit.ly/3NS6RrQ

Existing Problems → Developed Solutions

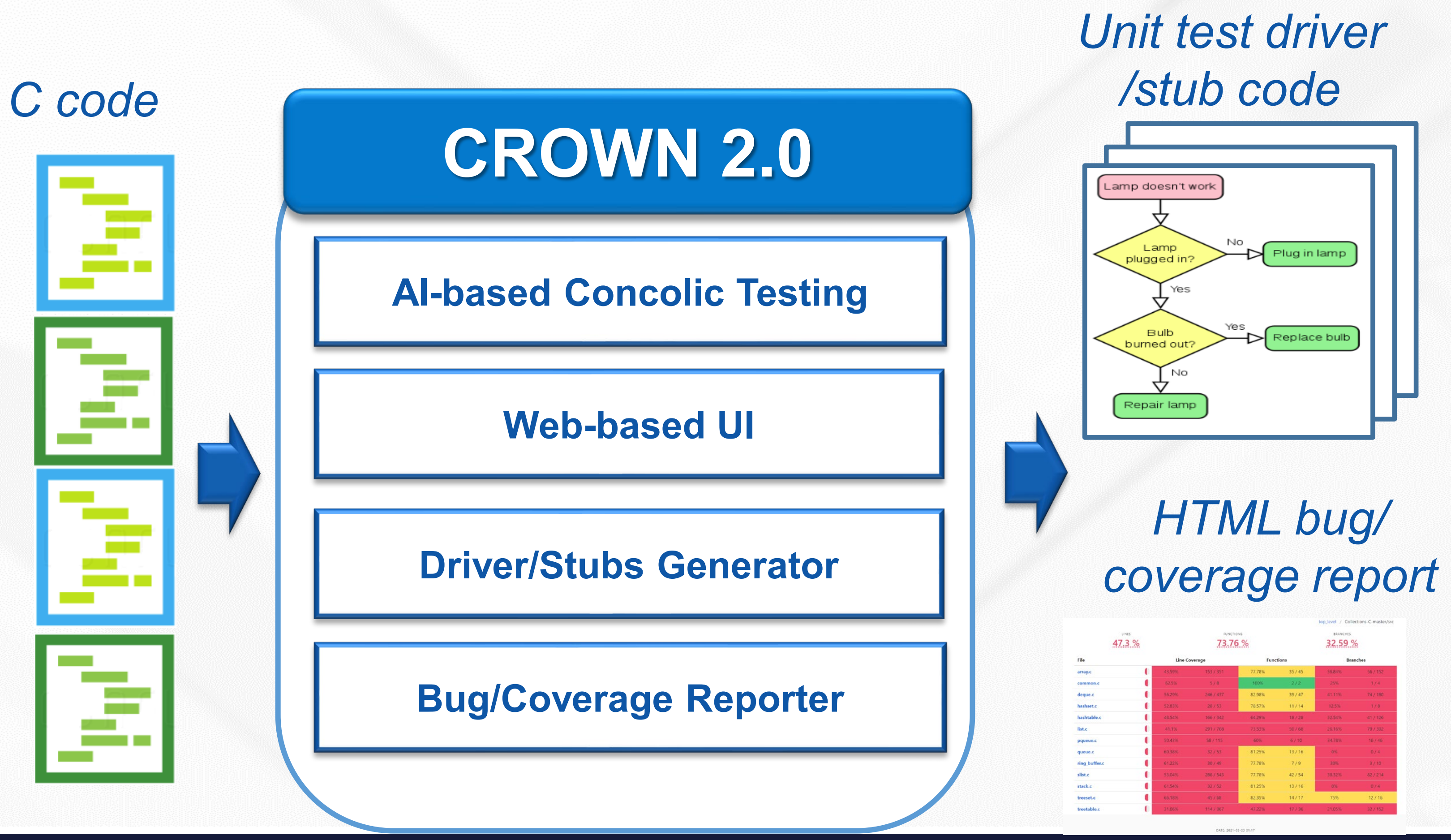




‘10-14 Project w/ Samsung Electronics	’15~20 Project w/ Hyundai/Mobis	‘18 Project w/ LIGnex1	’20 Project w/ Natl. Security Research Inst.
Detected dozens of crash bugs in the comm. firmware	Achieved 90% branch cov. and reduced 80% of labor cost by using auto. testing tech.	Detected several SW bugs in the 10 programs in the battleships	Detected SW bugs in the software in the security equipment

CROWN 2.0: Comercial Automated. Unit Testing Tool

CROWN 2.0 is a fully automated software test solution that significantly increases bug detection power and reduces testing cost for embedded C programs



Product Features

- **Automatically build stub and driver code**
- **Automatic test case generation based on AI-based Concolic testing**
- **Code-coverage report and analysis**
- **Test execution playback to help in debugging crash bugs**

Highlights

- **Eliminate need to write test code manually**
- **100% automated test code generation**
- **High-quality unit test generation**
- **Integrated code-coverage analysis**