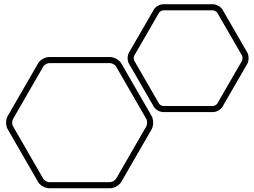
MANCHESTER 1824

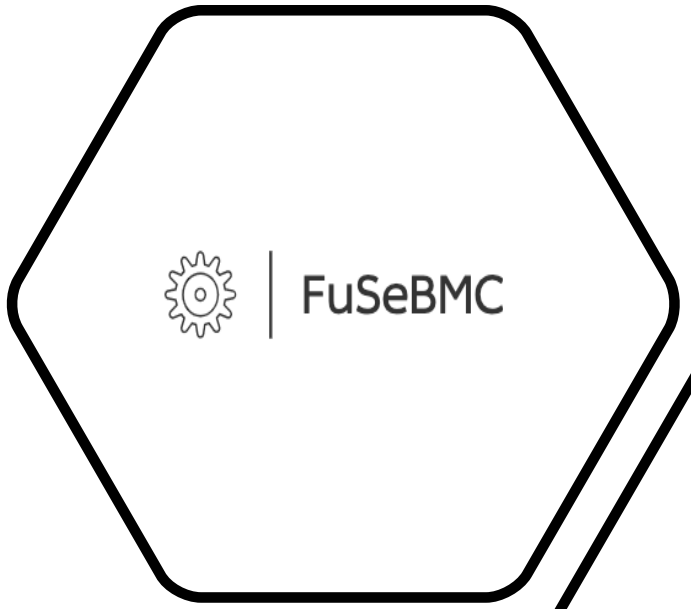
The University of Manchester

FuSeBMC

FuSeBMC: A White-Box Fuzzer for Finding Security Vulnerabilities in C Programs

Kaled Alshmrany
Kaled.alshmrany@manchester.ac.uk
Supervisor: Dr. Lucas Cordeiro
Co-Supervisor: Dr. Ning Zhang





The Outline

- FuSeBMC Team
- Aims
- FuSeBMC framework
- Evaluation
- Experiments
- Results
- Software Project
- Awards & Papers

FuSeBMC Team

Mr. Kaled Alshmrany

Dr. Lucas Cordeiro

FuSeBMC

Mr. Mikhail R. Gadelha

Mr. Rafael S. Menezes

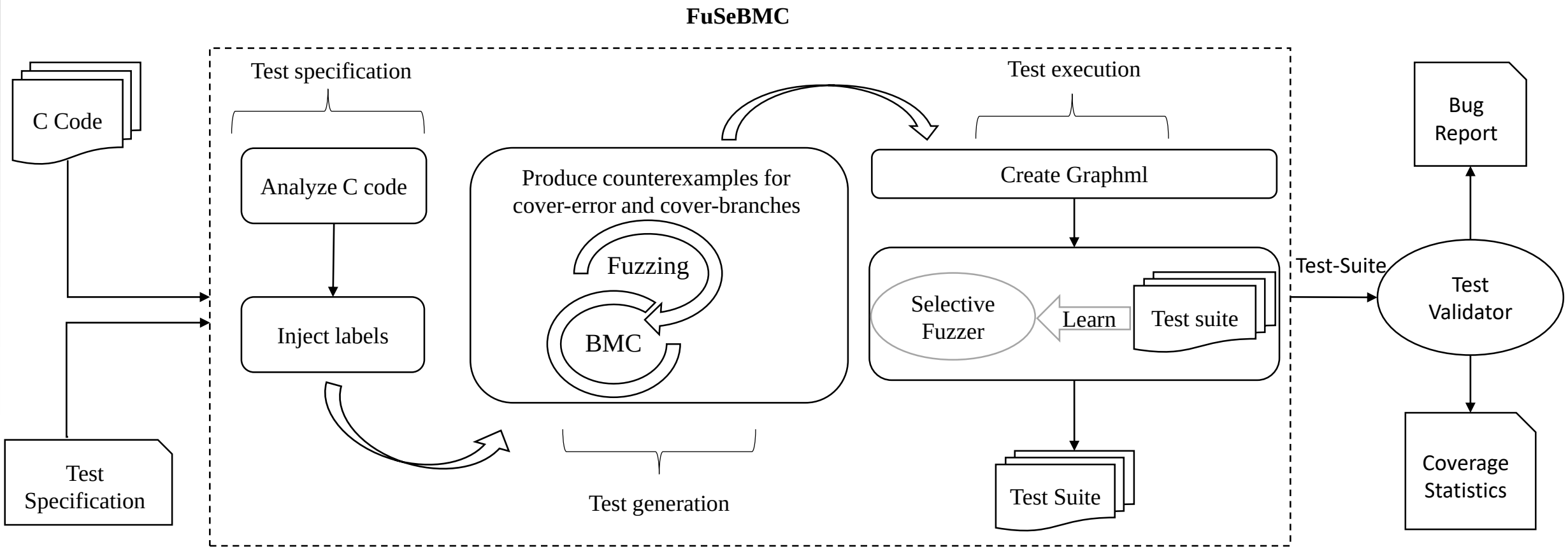
Research Aim

Design an effective tool for detecting vulnerabilities and achieving a high coverage

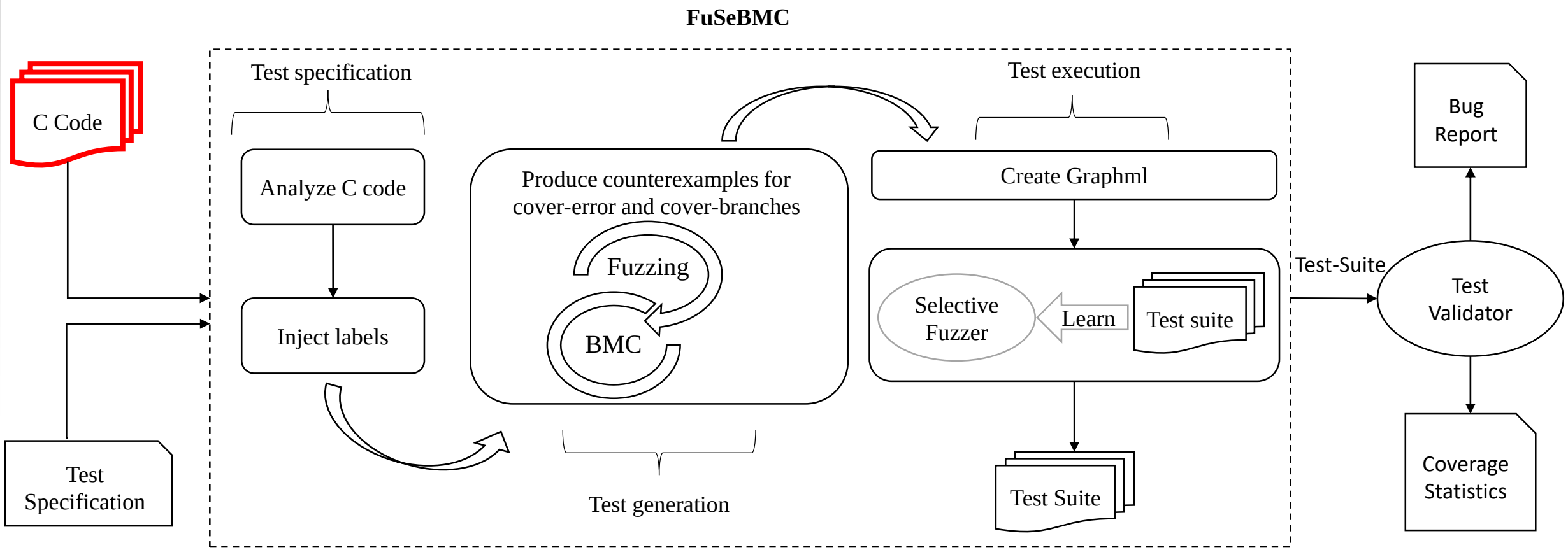
FuSeBMC

We describe and evaluate a novel a white-box fuzzer named FuSeBMC, which combines fuzzing and symbolic execution, and applies Bounded Model Checking (BMC) to find security vulnerabilities in C programs.

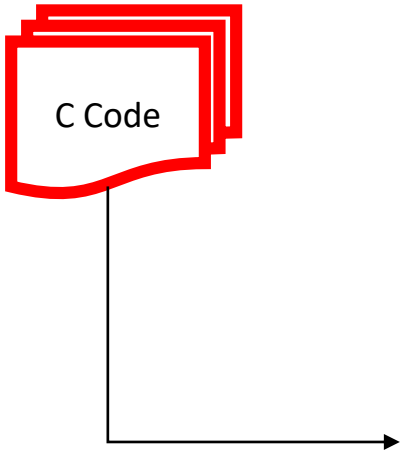
FuSeBMC Framework



FuSeBMC Framework

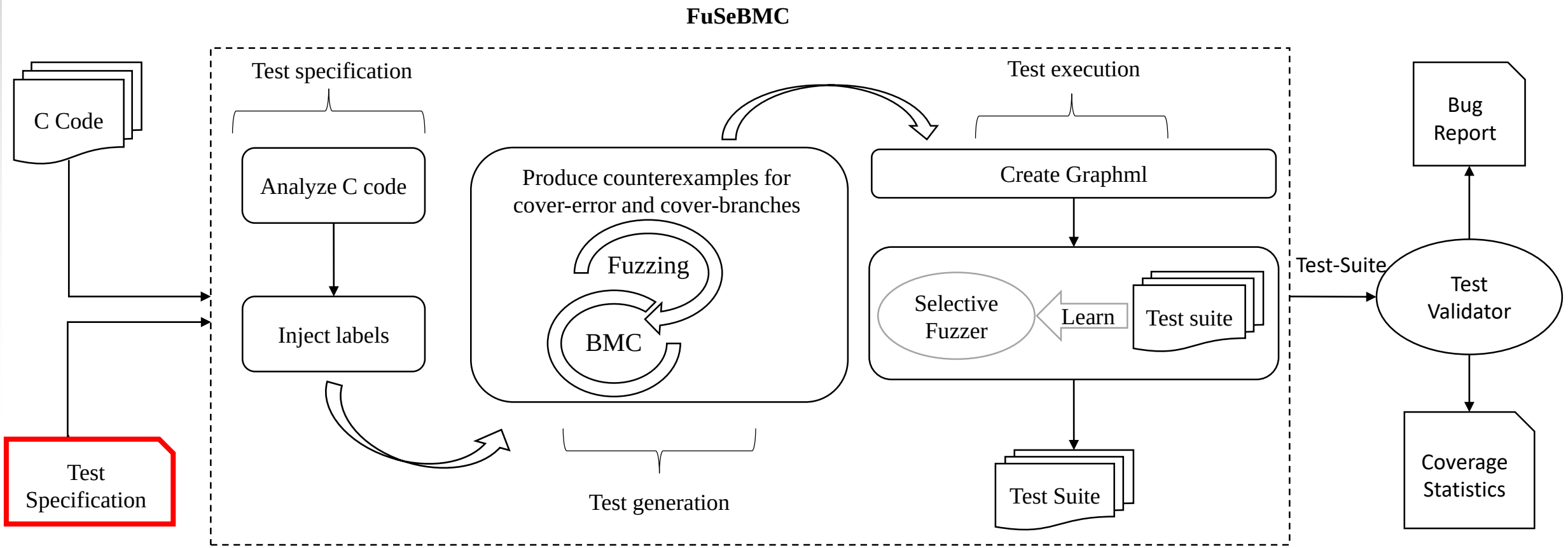


FuSeBMC Framework




```c
int main()
{
    int a = __VERIFIER_nondet_int();
    int b = __VERIFIER_nondet_int();
    int c = a + b;

    if (a > 0)
        return 1;
    else
        reach_error();

return 0;
}
```

FuSeBMC Framework



FuSeBMC Framework

Test Specification

Coverage-Error-Call

The test suite contains at least one test that executes function “reach_error( )”

Coverage-Branches

The test suite contains tests such that all branches of the program are executed

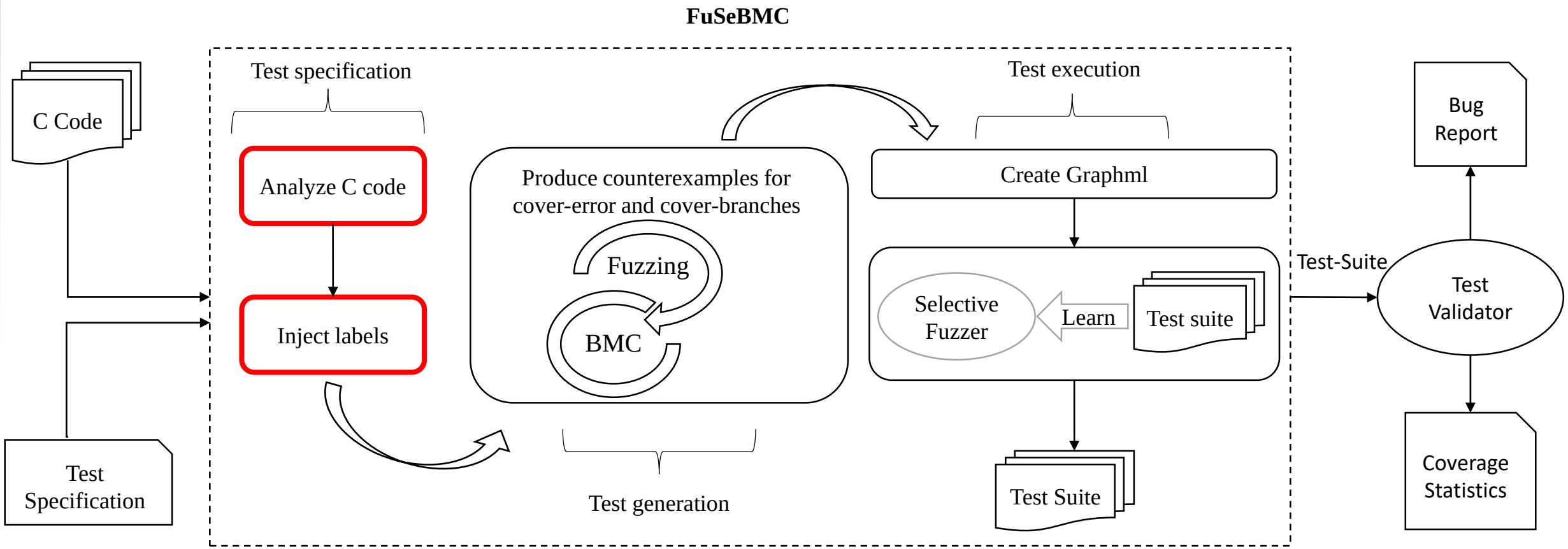
Formula of Error-Call:

```
COVER( init(main()), FQL(COVER EDGES(@CALL(reach_error))) )
```

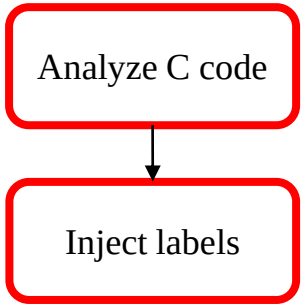
Formula of Cover-Branches:

```
COVER( init(main()), FQL(COVER EDGES(@DECISIONEDGE)) )
```

FuSeBMC Framework

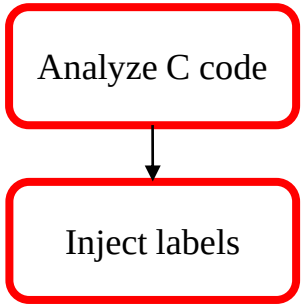


FuSeBMC Framework



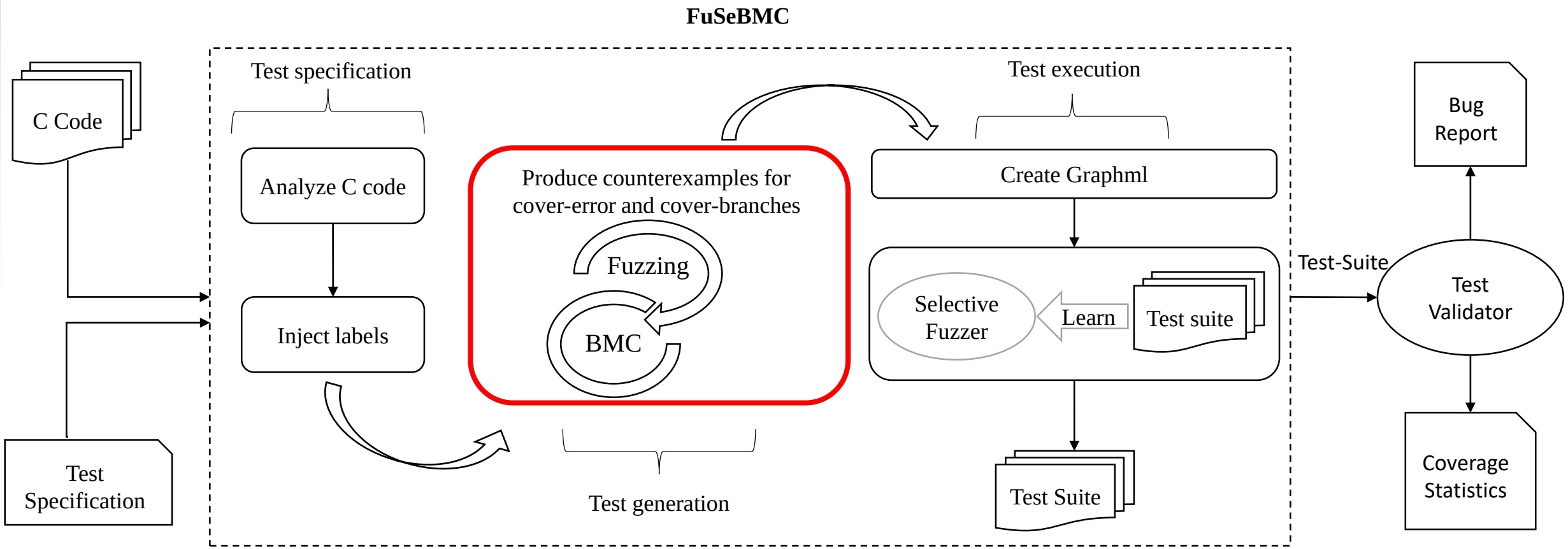

```
int main()
{
    int a = __VERIFIER_nondet_int();
    int b = __VERIFIER_nondet_int();
    int c = a + b;

    if (a > 0)
    {
        GOAL_2:;
        return 1;
    }
    else
    {
        GOAL_3:;
        reach_error();
    }

GOAL_1:;
return 0;
}
```

FuSeBMC Framework

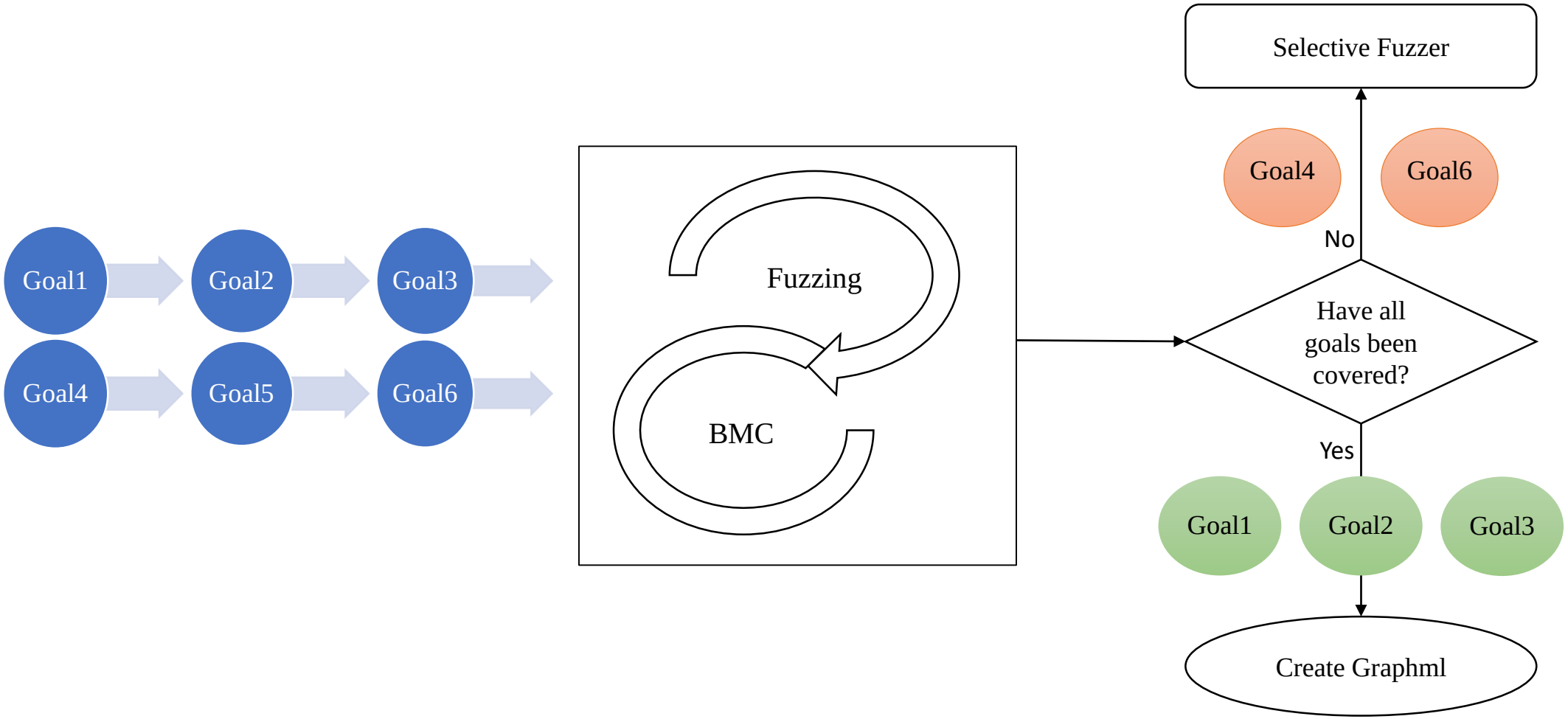



```
int main()
{
    int a = __VERIFIER_nondet_int();
    int b = __VERIFIER_nondet_int();
    int c = a + b;

    if (a > 0)
    {
        GOAL_2:;
        return 1;
    }
    else
    {
        GOAL_3:;
        reach_error();
    }

GOAL_1:;
return 0;
}
```

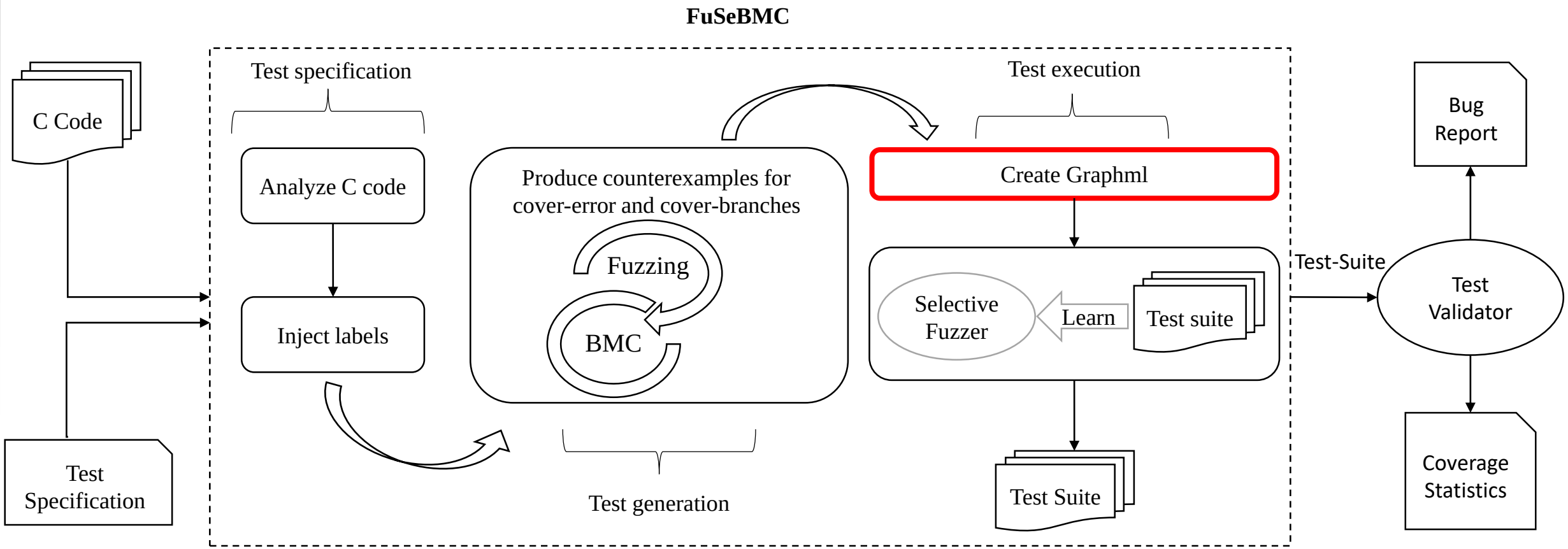
FuSeBMC Framework



FuSeBMC Framework



FuSeBMC Framework



FuSeBMC Framework

Create Graphml

GOAL_1.graphml

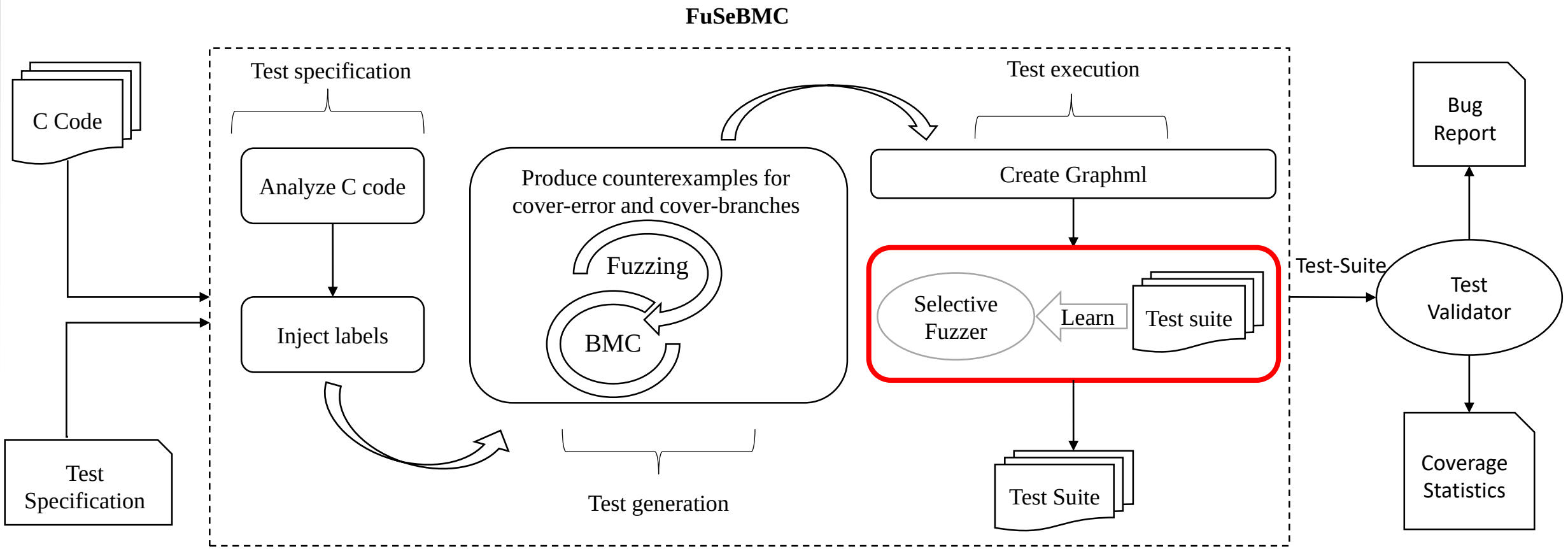
```
<?xml version="1.0" encoding="utf-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <key id="frontier" attr.name="isFrontierNode" attr.type="boolean" for="node">
    <default>false</default>
  </key>
  <key id="violation" attr.name="isViolationNode" attr.type="boolean" for="node">
    <default>false</default>
  </key>
  <key id="entry" attr.name="isEntryNode" attr.type="boolean" for="node">
    <default>false</default>
  </key>
  <key id="sink" attr.name="isSinkNode" attr.type="boolean" for="node">
    <default>false</default>
  </key>
  <key id="cyclehead" attr.name="cyclehead" attr.type="boolean" for="node">
    <default>false</default>
  </key>
  <key id="sourcecodelang" attr.name="sourcecodeLanguage" attr.type="string" for="graph"/>
  <key id="programfile" attr.name="programfile" attr.type="string" for="graph"/>
  <key id="programhash" attr.name="programhash" attr.type="string" for="graph"/>
  <key id="creationtime" attr.name="creationtime" attr.type="string" for="graph"/>
  <key id="specification" attr.name="specification" attr.type="string" for="graph"/>
  <key id="architecture" attr.name="architecture" attr.type="string" for="graph"/>
```

FuSeBMC Framework

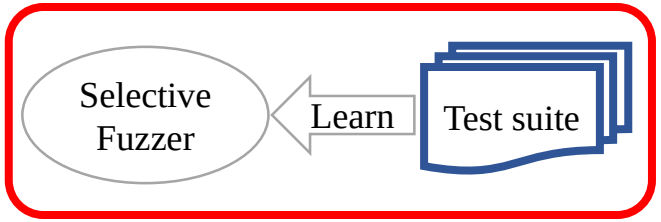
Create Graphml


```
<edge id="E2" source="N2" target="N3">
  <data key="startline">3</data>
  <data key="assumption">a = -2147483647;</data>
  <data key="threadId">0</data>
</edge>

<edge id="E4" source="N4" target="N5">
  <data key="startline">4</data>
  <data key="assumption">b = 0;</data>
  <data key="threadId">0</data>
</edge>
```

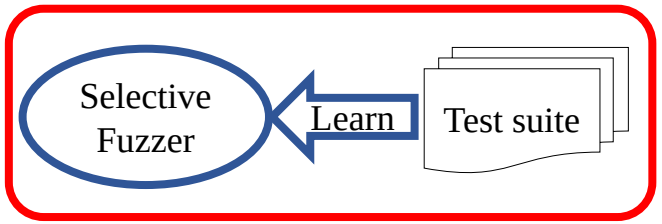
FuSeBMC Framework

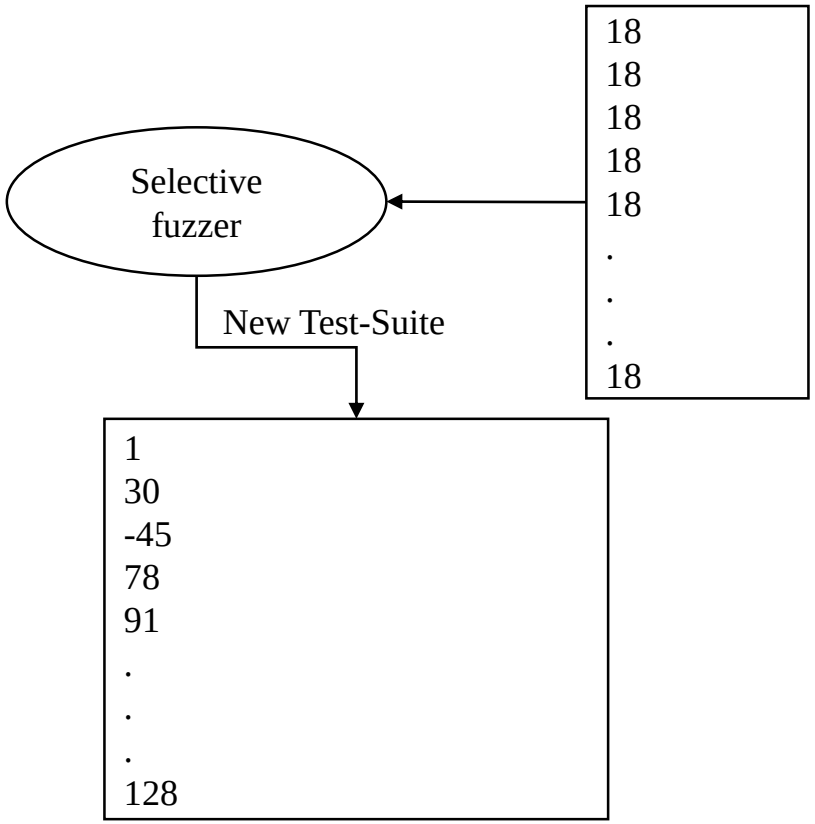


FuSeBMC Framework

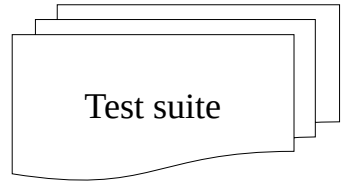



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?><!DOCTYPE testcase PUBLIC "+//IDN sosy-lab.org//DTD test-format testcase
1.0//EN" "https://sosy-lab.org/test-format/testcase-1.0.dtd">
<testcase>
    <input>-2147483647</input>
    <input>0</input>
</testcase>
```





The selective fuzzer will produce random number N (1000 times) based on the information we got from Fuzzer/BMC

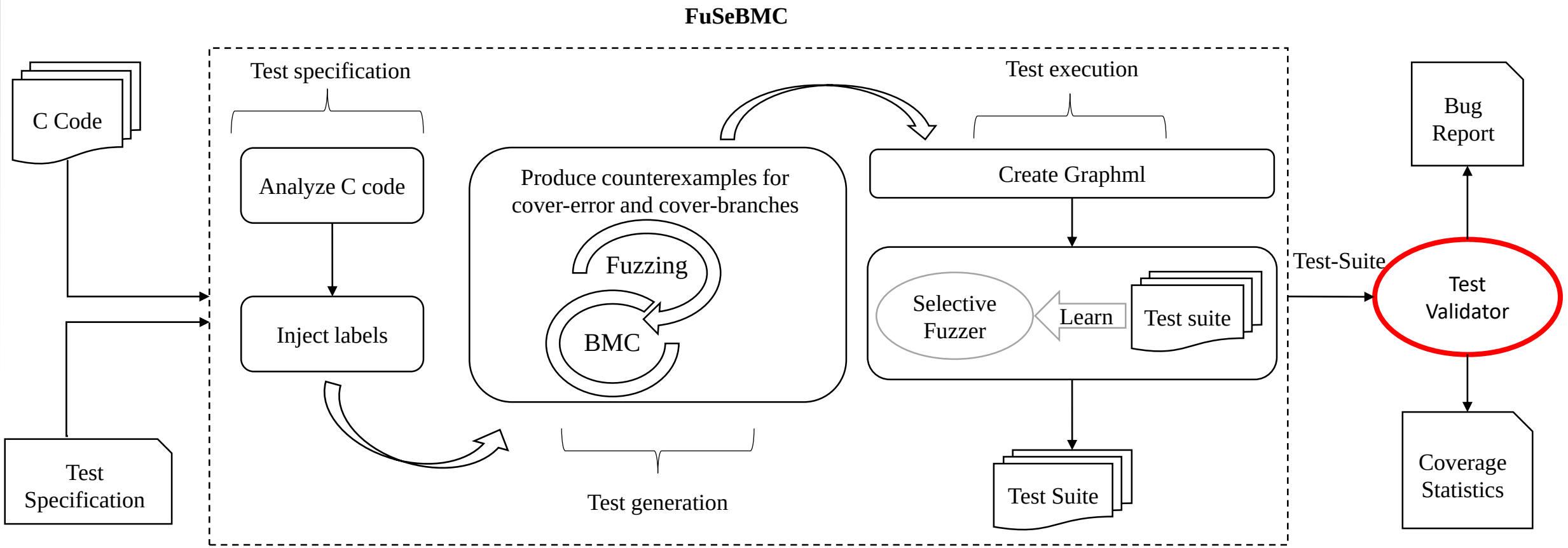


One example of Test-Suite

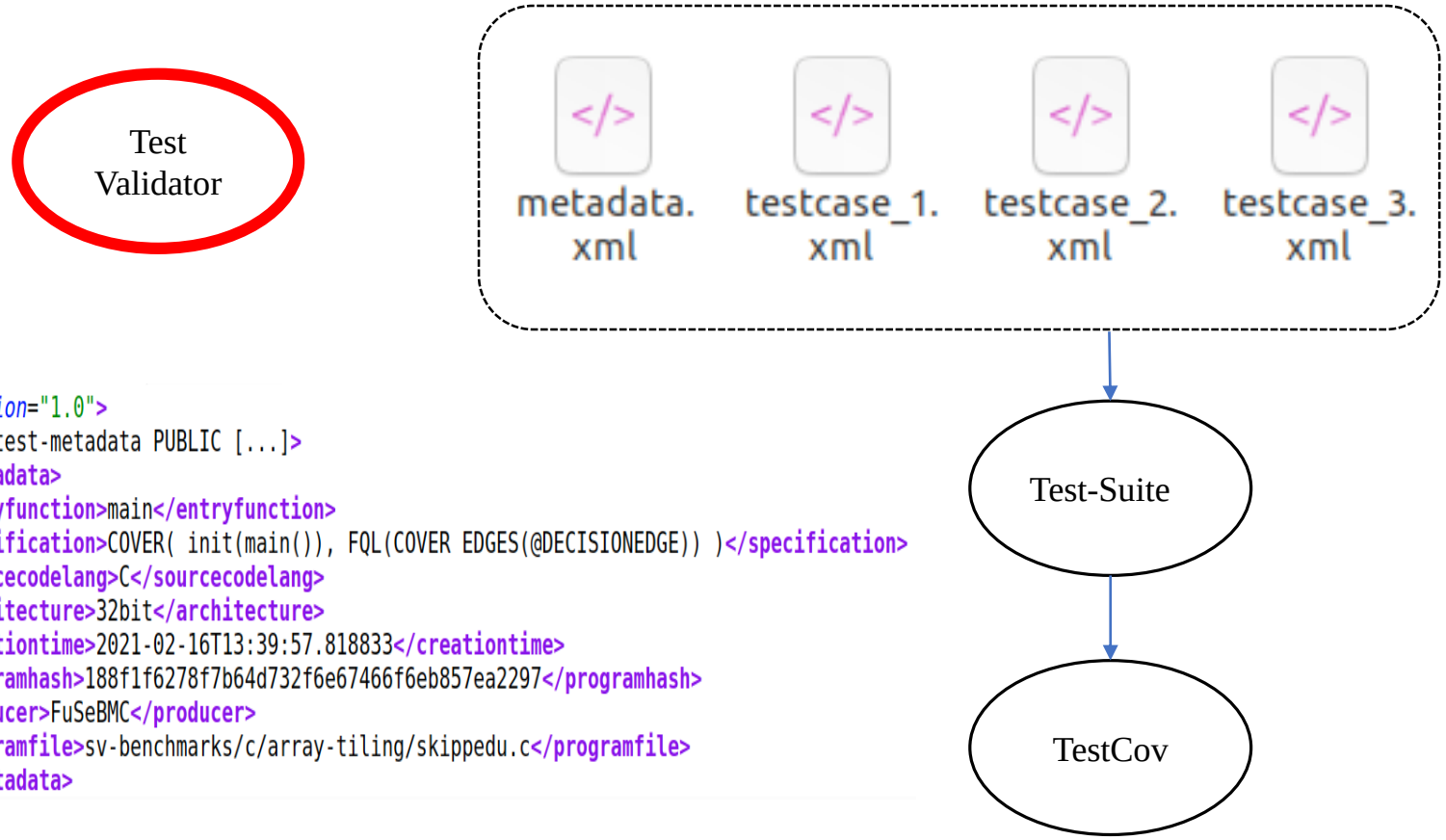
18
18
18
18
18
.
.
.
18

We assumed that the Fuzzer/BMC passes the values 18 (1000 times)

FuSeBMC Framework

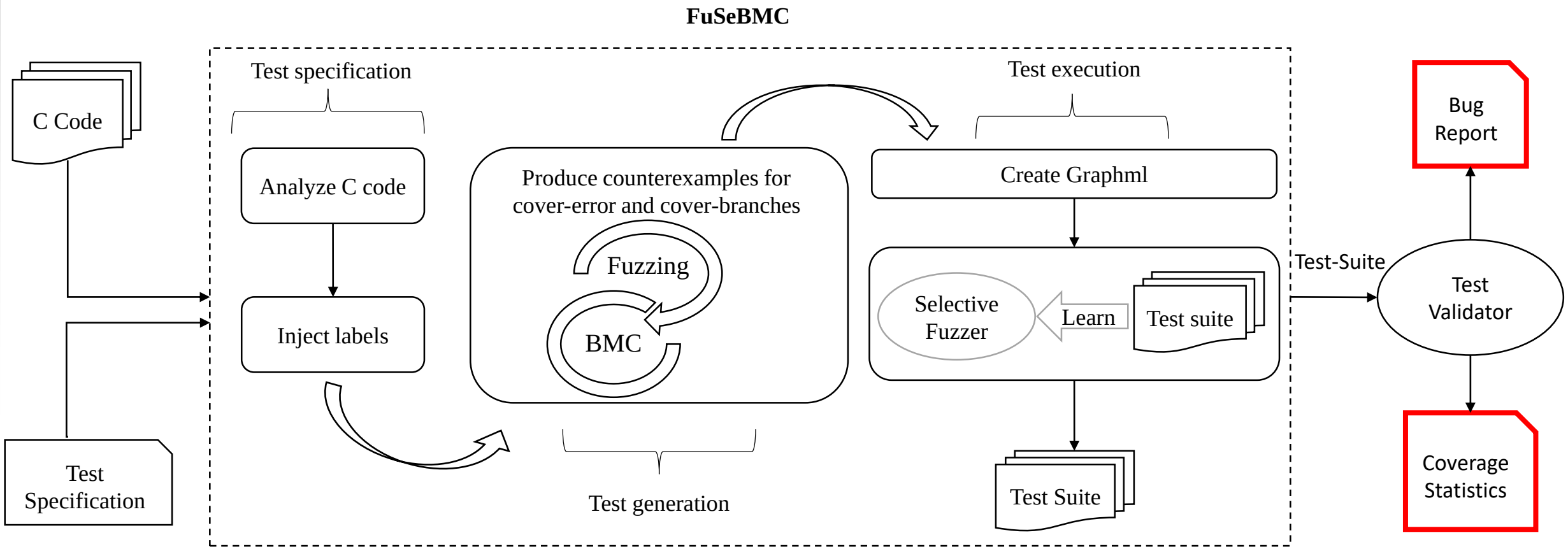


FuSeBMC Framework

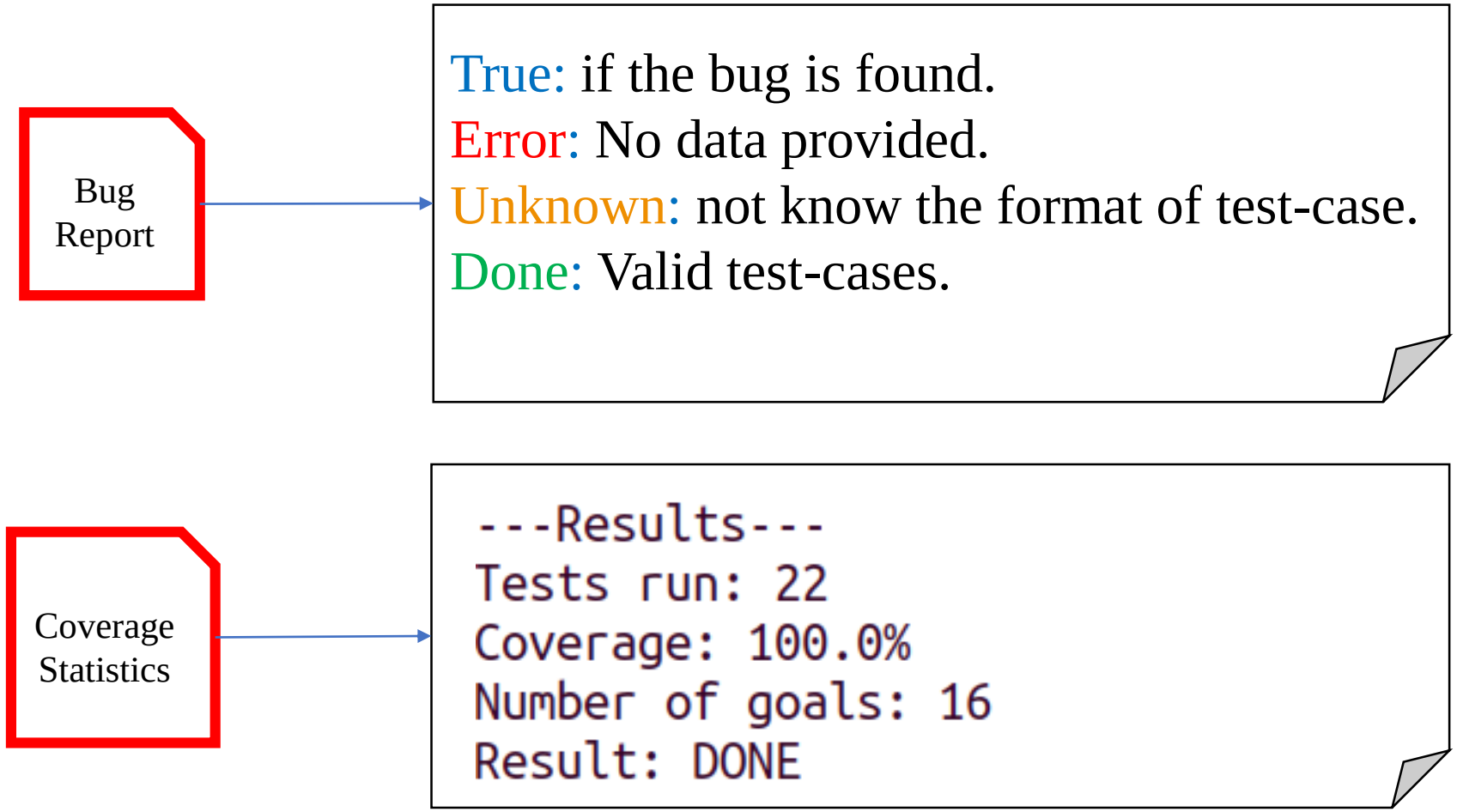



```
<xml version="1.0">
<DOCTYPE test-metadata PUBLIC [...]>
<test-metadata>
    <entryfunction>main</entryfunction>
    <specification>COVER( init(main()), FQL(COVER EDGES(@DECISIONEDGE)) )</specification>
    <sourcecodelang>C</sourcecodelang>
    <architecture>32bit</architecture>
    <creationtime>2021-02-16T13:39:57.818833</creationtime>
    <programhash>188f1f6278f7b64d732f6e67466f6eb857ea2297</programhash>
    <producer>FuSeBMC</producer>
    <programfile>sv-benchmarks/c/array-tiling/skippedu.c</programfile>
</test-metadata>
```

FuSeBMC Framework

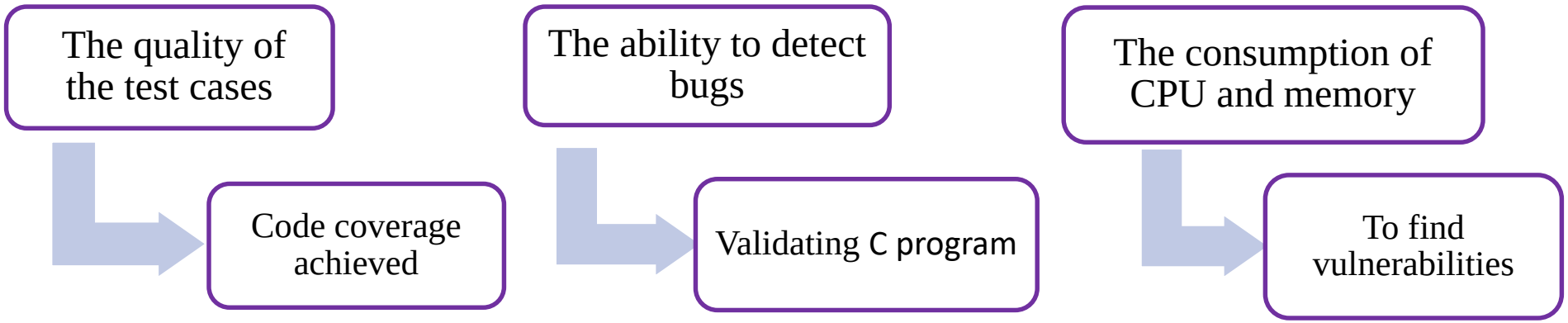


FuSeBMC Framework



Evaluation

- Our proposed approach, "FuSeBMC" can be evaluated in three criteria :



Evaluation

- Our evaluation aims to answer three main questions (goals):

Q1- Could FuSeBMC detect effectively and efficiently vulnerabilities? → **Obj1**

Q2- Could FuSeBMC achieve high coverage compared with the state-of-the-art tools? → **Obj2**

Q3- Could FuSeBMC reduce the consumption of CPU and memory compared with the state-of-the-art tools? → **Obj3**

Experiments

- We conducted experiments with FuSeBMC on the benchmark of the 3rd Intl. Competition on Software Testing (Test-Comp 2021).
- The competition has two categories Error Coverage and Branch Coverage

Error Coverage	is to show the abilities to discover bugs
Branch Coverage	is to cover as many branches as possible

Results of the Error Coverage

Cover-Error	Task-Num	FuSeBMC	CMA-ES Fuzz	CoVeriTest	HybridTiger	KLEE	Legion	LibKluzzer	PRTest	Symbiotic	Tracer-X	VeriFuzz
ReachSafety-Arrays	100	93	0	59	69	88	67	96	11	73	75	95
ReachSafety-BitVectors	10	**10**	0	8	6	9	0	9	5	8	7	9
ReachSafety-ControlFlow	32	8	0	8	8	10	0	11	0	7	9	9
ReachSafety-ECA	18	8	0	2	1	14	0	11	0	15	2	16
ReachSafety-Floats	33	**32**	0	16	22	6	0	30	3	0	0	30
ReachSafety-Heap	57	45	0	37	38	46	0	47	9	47	44	47
ReachSafety-Loops	158	131	0	35	53	96	4	138	102	82	78	136
ReachSafety-Recursive	20	**19**	0	0	5	16	0	17	1	17	14	13
ReachSafety-Sequentialized	107	**101**	0	61	93	86	0	83	0	79	57	99
ReachSafety-XCSP	59	**53**	0	46	52	37	0	3	0	41	31	25
SoftwareSystems -BusyBox-MemSafety	11	0	0	0	0	0	0	0	0	0	0	0
SoftwareSystems-DeviceDriversLinux64-ReachSafety	2	0	0	0	0	0	0	0	0	0	0	0
Sum	607	500	0	272	347	408	71	442	131	369	317	479
Error		405	0	225	266	339	35	359	79	314	246	385

Results of the Error Coverage

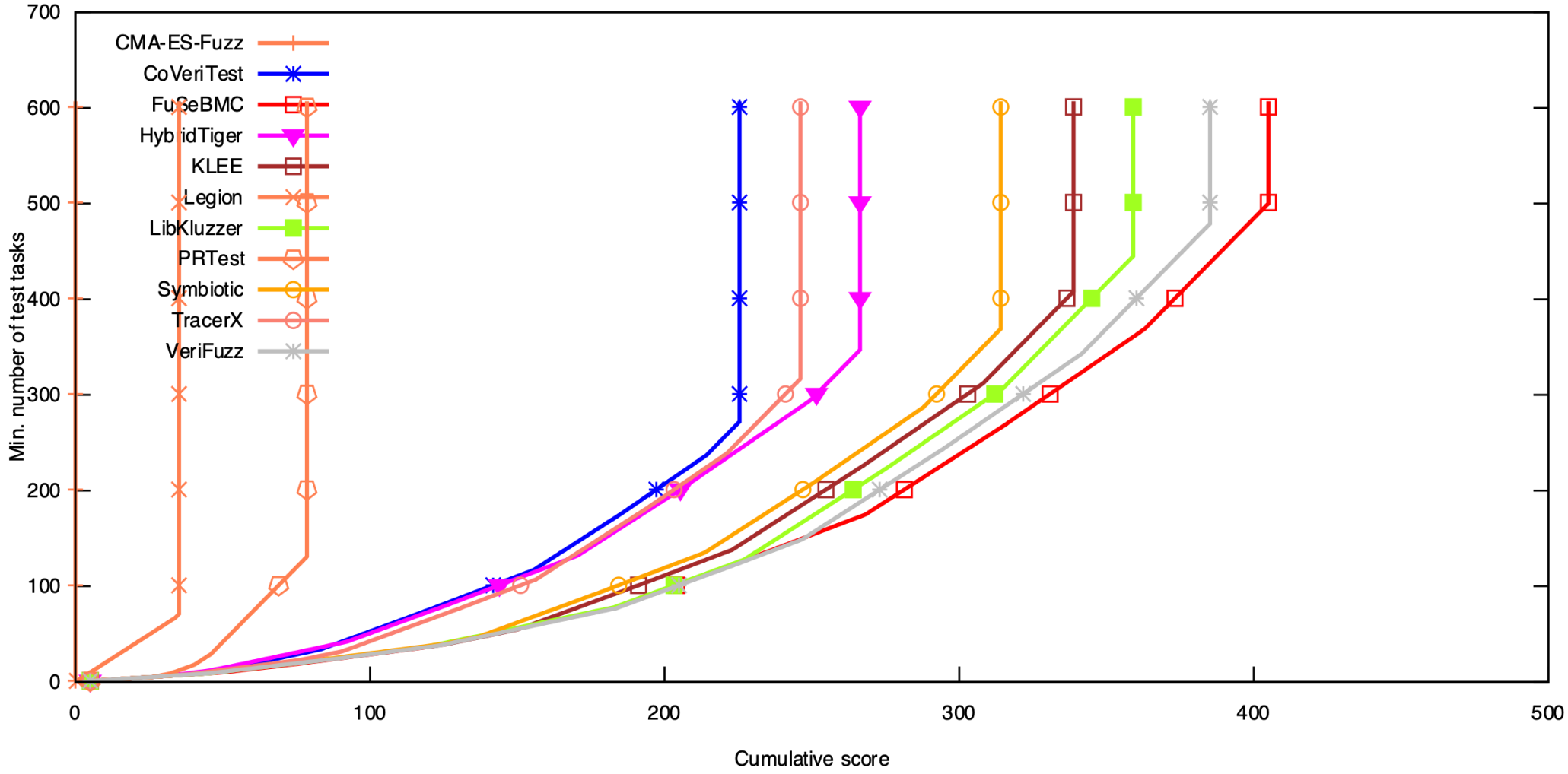
Cover-Error	Task-Num	FuSeBMC	CMA-ES Fuzz	CoVeriTest	HybridTiger	KLEE	Legion	LibKluzzer	PRTest	Symbiotic	Tracer-X	VeriFuzz
ReachSafety-Arrays	100	93	0	59	69	88	67	96	11	73	75	95
ReachSafety-BitVectors	10	**10**	0	8	6	9	0	9	5	8	7	9
ReachSafety-ControlFlow	32	8	0	8	8	10	0	11	0	7	9	9
ReachSafety-ECA	18	8	0	2	1	14	0	11	0	15	2	16
ReachSafety-Floats	33	**32**	0	16	22	6	0	30	3	0	0	30
ReachSafety-Heap	57	45	0	37	38	46	0	47	9	47	44	47
ReachSafety-Loops	158	131	0	35	53	96	4	138	102	82	78	136
ReachSafety-Recursive	20	**19**	0	0	5	16	0	17	1	17	14	13
ReachSafety-Sequentialized	107	**101**	0	61	93	86	0	83	0	79	57	99
ReachSafety-XCSP	59	**53**	0	46	52	37	0	3	0	41	31	25
SoftwareSystems -BusyBox-MemSafety	11	0	0	0	0	0	0	0	0	0	0	0
SoftwareSystems-DeviceDriversLinux64-ReachSafety	2	0	0	0	0	0	0	0	0	0	0	0
Sum	607	500	0	272	347	408	71	442	131	369	317	479
Error		405	0	225	266	339	35	359	79	314	246	385

Results of the Error Coverage

Obj1

Cover-Error	Task-Num	FuSeBMC	CMA-ES Fuzz	CoVeriTest	HybridTiger	KLEE	Legion	LibKluzzer	PRTest	Symbiotic	Tracer-X	VeriFuzz
ReachSafety-Arrays	100	93	0	59	69	88	67	96	11	73	75	95
ReachSafety-BitVectors	10	**10**	0	8	6	9	0	9	5	8	7	9
ReachSafety-ControlFlow	32	8	0	8	8	10	0	11	0	7	9	9
ReachSafety-ECA	18	8	0	2	1	14	0	11	0	15	2	16
ReachSafety-Floats	33	**32**	0	16	22	6	0	30	3	0	0	30
ReachSafety-Heap	57	45	0	37	38	46	0	47	9	47	44	47
ReachSafety-Loops	158	131	0	35	53	96	4	138	102	82	78	136
ReachSafety-Recursive	20	**19**	0	0	5	16	0	17	1	17	14	13
ReachSafety-Sequentialized	107	**101**	0	61	93	86	0	83	0	79	57	99
ReachSafety-XCSP	59	**53**	0	46	52	37	0	3	0	41	31	25
SoftwareSystems -BusyBox-MemSafety	11	0	0	0	0	0	0	0	0	0	0	0
SoftwareSystems-DeviceDriversLinux64-ReachSafety	2	0	0	0	0	0	0	0	0	0	0	0
Sum	607	500	0	272	347	408	71	442	131	369	317	479
Error		405	0	225	266	339	35	359	79	314	246	385

Results of the Error Coverage



Results of the Branch Coverage

Cover-Branches	Task-Num	FuSeBMC	CMA-ES Fuzz	CoVeriTest	HybridTiger	KLEE	Legion	LibKluzzer	PRTest	Symbiotic	Tracer-X	VeriFuzz
ReachSafety-Arrays	400	284	139	229	225	96	195	296	119	226	223	295
ReachSafety-BitVectors	62	37	23	39	13	28	29	40	27	37	37	38
ReachSafety-ControlFlow	67	15	4	16	3	8	8	16	5	18	15	19
ReachSafety-ECA	29	5	0	6	2	7	3	10	2	10	7	12
ReachSafety-Floats	226	103	51	99	84	16	64	90	41	50	48	99
ReachSafety-Heap	143	88	19	79	74	81	69	90	40	84	86	86
ReachSafety-Loops	581	412	152	402	338	274	271	419	252	383	385	424
ReachSafety-Recursive	53	36	19	31	31	18	21	36	9	38	34	35
ReachSafety-Sequentialized	82	62	0	61	39	26	1	55	8	36	41	71
ReachSafety-XCSP	119	97	0	80	80	81	2	80	79	93	69	88
ReachSafety-Combinations	210	15	0	31	8	82	18	139	2	135	99	180
SoftwareSystems -BusyBox-MemSafety	72	1	0	5	4	6	0	6	4	7	4	8
SoftwareSystems-DeviceDriversLinux64-ReachSafety	290	35	13	60	6	25	56	58	16	44	57	57
SoftwareSystems-SQLite-MemSafety	1	0	0	0	0	0	0	0	0	0	0	0
Termination-MainHeap	231	202	138	193	189	119	166	199	51	178	186	204
Sum	2566	1391	558	1331	1096	867	902	1534	654	1338	1291	1615
BR		1161	411	1128	860	784	651	1292	519	1169	1087	13

Results of the Branch Coverage

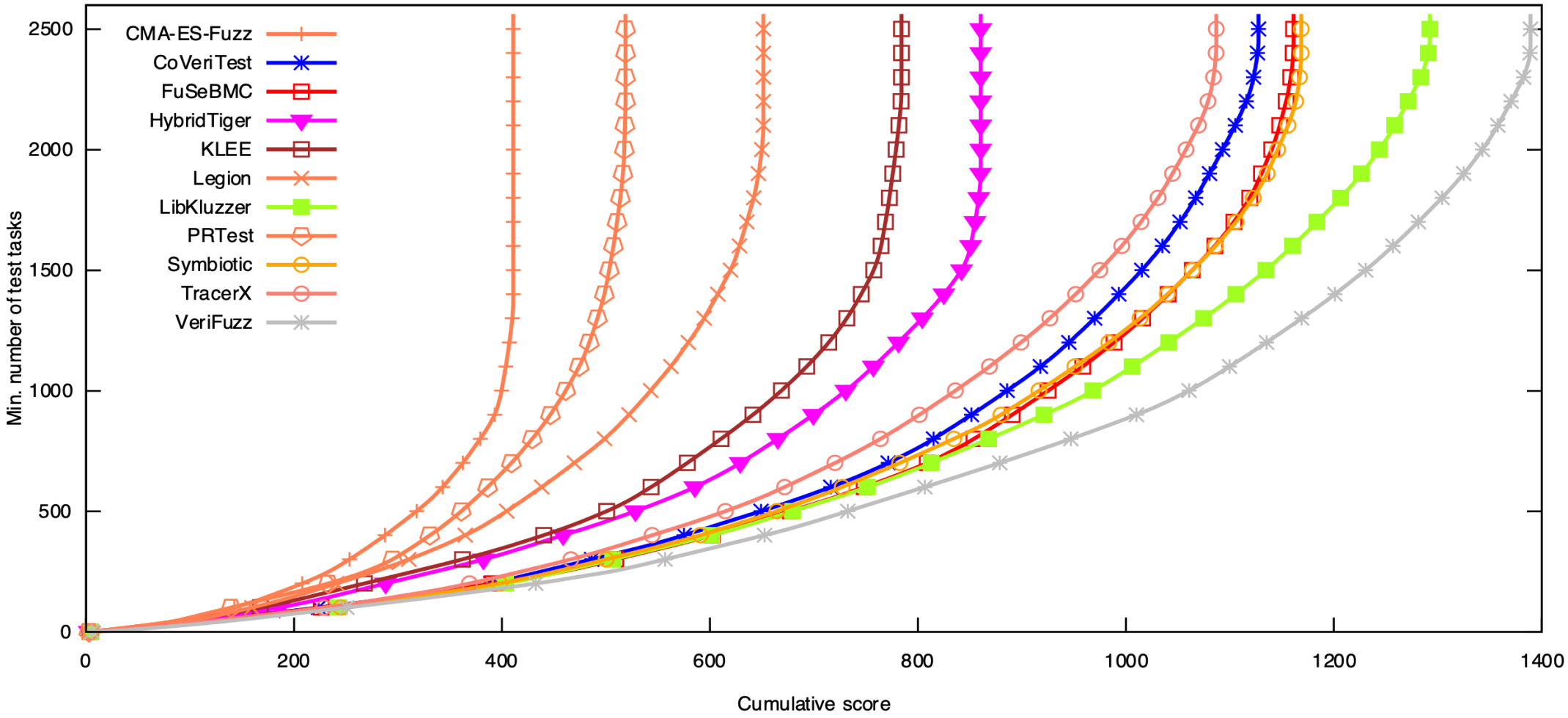
Cover-Branches	Task-Num	FuSeBMC	CMA-ES Fuzz	CoVeriTest	HybridTiger	KLEE	Legion	LibKluzzer	PRTest	Symbiotic	Tracer-X	VeriFuzz
ReachSafety-Arrays	400	284	139	229	225	96	195	296	119	226	223	295
ReachSafety-BitVectors	62	37	23	39	13	28	29	40	27	37	37	38
ReachSafety-ControlFlow	67	15	4	16	3	8	8	16	5	18	15	19
ReachSafety-ECA	29	5	0	6	2	7	3	10	2	10	7	12
ReachSafety-Floats	226	103	51	99	84	16	64	90	41	50	48	99
ReachSafety-Heap	143	88	19	79	74	81	69	90	40	84	86	86
ReachSafety-Loops	581	412	152	402	338	274	271	419	252	383	385	424
ReachSafety-Recursive	53	36	19	31	31	18	21	36	9	38	34	35
ReachSafety-Sequentialized	82	62	0	61	39	26	1	55	8	36	41	71
ReachSafety-XCSP	119	97	0	80	80	81	2	80	79	93	69	88
ReachSafety-Combinations	210	15	0	31	8	82	18	139	2	135	99	180
SoftwareSystems -BusyBox-MemSafety	72	1	0	5	4	6	0	6	4	7	4	8
SoftwareSystems-DeviceDriversLinux64-ReachSafety	290	35	13	60	6	25	56	58	16	44	57	57
SoftwareSystems-SQLite-MemSafety	1	0	0	0	0	0	0	0	0	0	0	0
Termination-MainHeap	231	202	138	193	189	119	166	199	51	178	186	204
Sum	2566	1391	558	1331	1096	867	902	1534	654	1338	1291	1615
BR		1161	411	1128	860	784	651	1292	519	1169	1087	1389

Results of the Branch Coverage

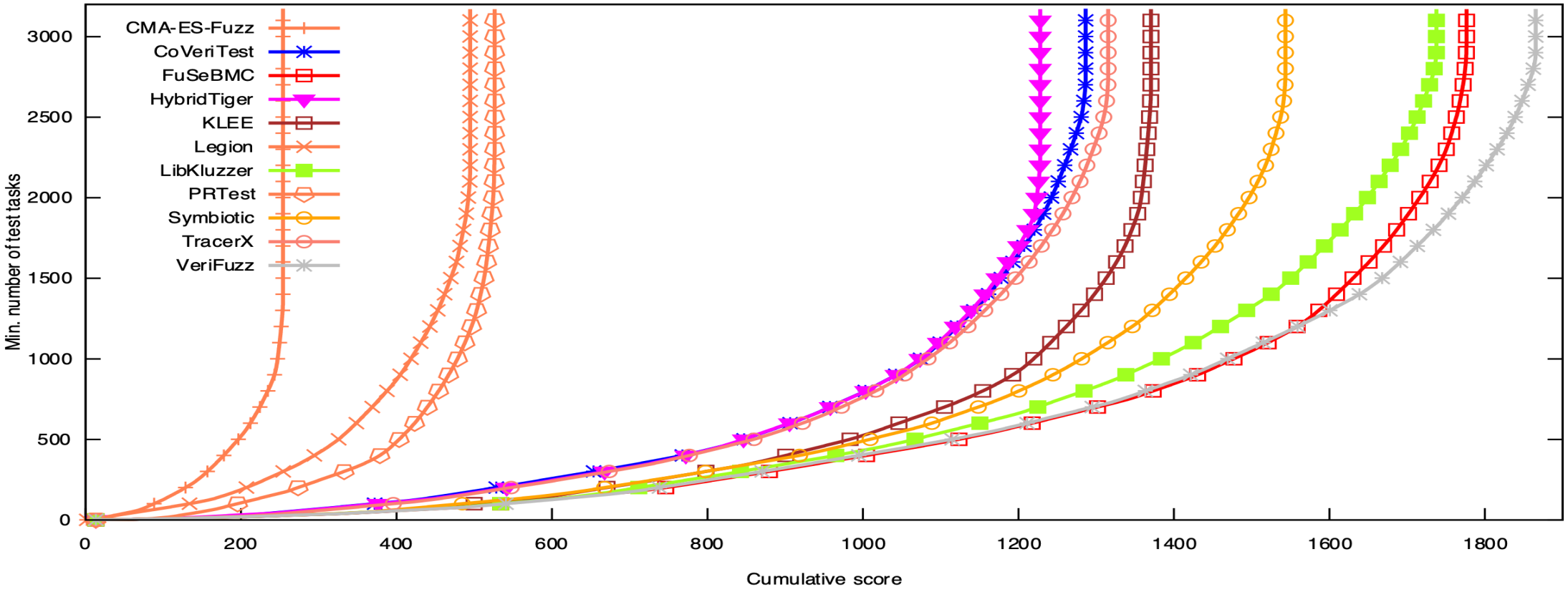
Obj2

Cover-Branches	Task-Num	FuSeBMC	CMA-ES Fuzz	CoVeriTest	HybridTiger	KLEE	Legion	LibKluzzer	PRTest	Symbiotic	Tracer-X	VeriFuzz
ReachSafety-Arrays	400	284	139	229	225	96	195	296	119	226	223	295
ReachSafety-BitVectors	62	37	23	39	13	28	29	40	27	37	37	38
ReachSafety-ControlFlow	67	15	4	16	3	8	8	16	5	18	15	19
ReachSafety-ECA	29	5	0	6	2	7	3	10	2	10	7	12
ReachSafety-Floats	226	103	51	99	84	16	64	90	41	50	48	99
ReachSafety-Heap	143	88	19	79	74	81	69	90	40	84	86	86
ReachSafety-Loops	581	412	152	402	338	274	271	419	252	383	385	424
ReachSafety-Recursive	53	36	19	31	31	18	21	36	9	38	34	35
ReachSafety-Sequentialized	82	62	0	61	39	26	1	55	8	36	41	71
ReachSafety-XCSP	119	97	0	80	80	81	2	80	79	93	69	88
ReachSafety-Combinations	210	15	0	31	8	82	18	139	2	135	99	180
SoftwareSystems -BusyBox-MemSafety	72	1	0	5	4	6	0	6	4	7	4	8
SoftwareSystems-DeviceDriversLinux64-ReachSafety	290	35	13	60	6	25	56	58	16	44	57	57
SoftwareSystems-SQLite-MemSafety	1	0	0	0	0	0	0	0	0	0	0	0
Termination-MainHeap	231	202	138	193	189	119	166	199	51	178	186	204
Sum	2566	1391	558	1331	1096	867	902	1534	654	1338	1291	1615
BR		1161	411	1128	860	784	651	1292	519	1169	1087	1389

Results of the Branch Coverage



The overall Results of Test-Comp 2020



Cover-Error & Branches	Task-Num	FuSeBMC	CMA-ES Fuzz	CoVeriTest	HybridTiger	KLEE	Legion	LibKluzzer	PRTest	Symbiotic	Tracer-X	VeriFuzz
OVERALL	3173	1776	254	1286	1228	1370	495	1738	526	1543	1315	1865

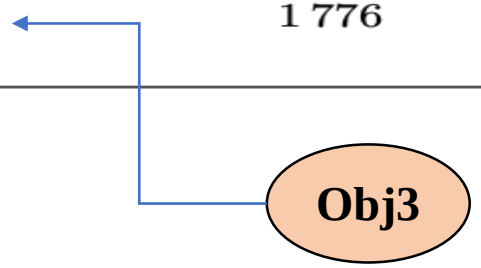
The Consumption of CPU and Memory

Rank	Test Generator	Quality (sp)	CPU Time (h)	CPU Energy (kWh)	Rank Measure (kJ/sp)
Green Testing					
1	TracerX	1 315	210	2.5	6.8
2	Klee	1 370	210	2.6	6.8
3	FuSeBMC	1 776	410	4.8	9.7
worst					51

Rank	Test Generator	Quality (sp)	CPU Time (h)	CPU Energy (kWh)	Rank Measure
*	LibKluzzer	1738	610	6.7	13.9
*	VeriFuzz	1865	640	8.1	15.6

The Consumption of CPU and Memory

Rank	Test Generator	Quality (sp)	CPU Time (h)	CPU Energy (kWh)	Rank Measure (kJ/sp)
Green Testing					
1	TracerX	1 315	210	2.5	6.8
2	Klee	1 370	210	2.6	6.8
3	FuSeBMC	1 776	410	4.8	9.7
worst					51



Rank	Test Generator	Quality (sp)	CPU Time (h)	CPU Energy (kWh)	Rank Measure
*	LibKluzzer	1738	610	6.7	13.9
*	VeriFuzz	1865	640	8.1	15.6

Software Project

- The FuSeBMC source code is written in C++ and it is available for download in GitHub. Also, the instructions for using the tool FuSeBMC are given in the file README.

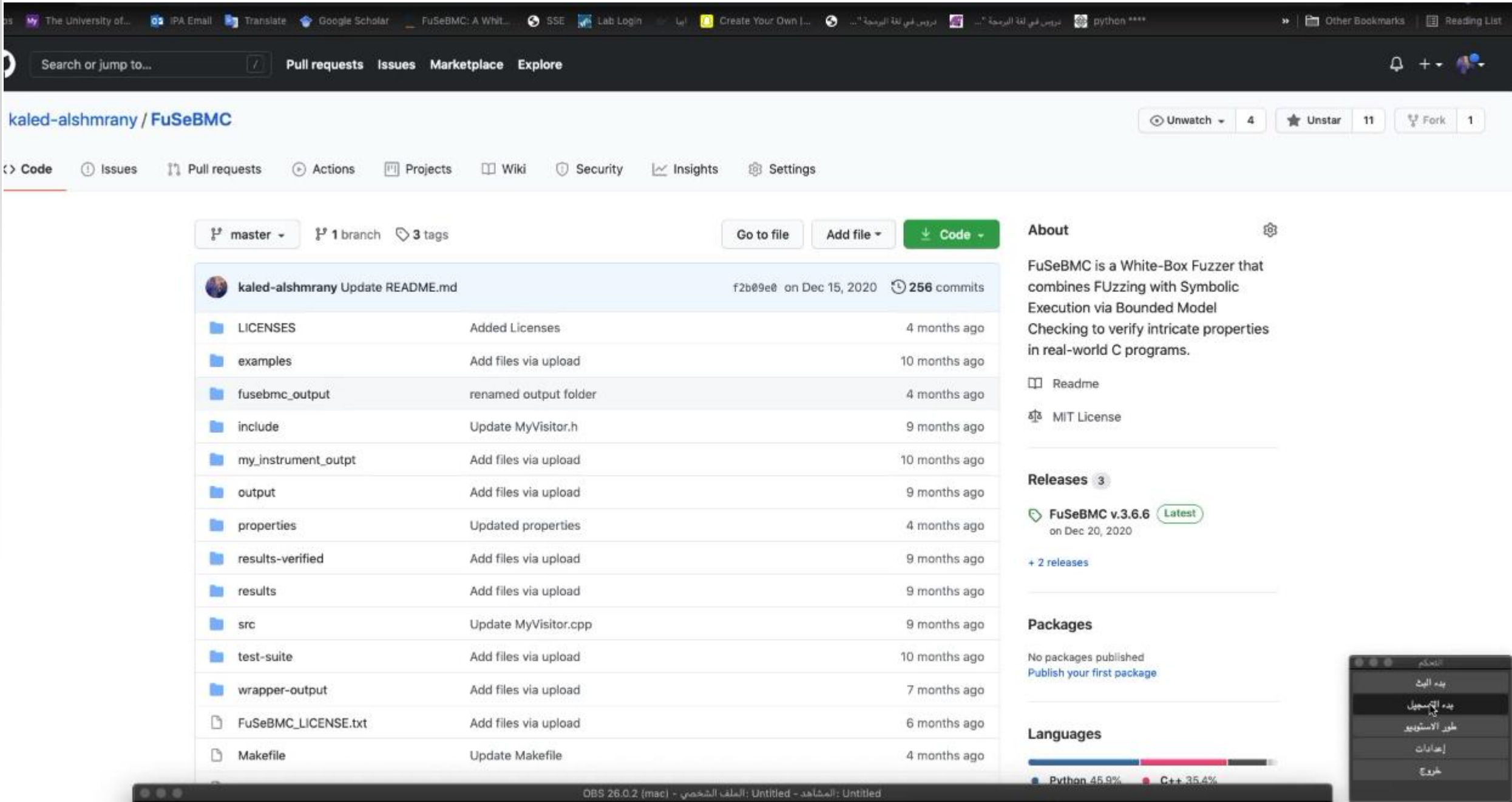
```
kaled@kaled-VirtualBox:~/Desktop/FuSeBMC_v3.6.6$ ./fusebmc.py -s incr -p properties
/coverage-branches.prp sv-benchmarks/c/array-tiling/skippedu.c
```

Awards & Papers

FuSeBMC received three significant awards from the 3rd International Competition on Software Testing (Test-Comp 2021) organised by the European Joint Conferences on Theory and Practice of Software (ETAPS).

- FuSeBMC got first place in the most critical category of Test-Comp: **Cover-Error** (find a test that covers a bug).
- FuSeBMC earned second place in Test-Comp's overall category, which includes **Cover-Branches** (find tests for branch coverage).
- FuSeBMC got the third place in ranking of **Consumption of CPU and Memory.**
- Accepted paper in Fundamental Approaches to Software Engineering – 24th International Conference, FASE 2021

Demo


Search or jump to...
Pull requests
Issues
Marketplace
Explore
kaled-alshmrany / FuSeBMC
Unwatch 4
Unstar 11
Fork 1
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
master
1 branch
3 tags
Go to file
Add file
Code
kaled-alshmrany Update README.md
f2b09e0 on Dec 15, 2020
256 commits
LICENSES Added Licenses 4 months ago
examples Add files via upload 10 months ago
fusebmc_output renamed output folder 4 months ago
include Update MyVisitor.h 9 months ago
my_instrument_outpt Add files via upload 10 months ago
output Add files via upload 9 months ago
properties Updated properties 4 months ago
results-verified Add files via upload 9 months ago
results Add files via upload 9 months ago
src Update MyVisitor.cpp 9 months ago
test-suite Add files via upload 10 months ago
wrapper-output Add files via upload 7 months ago
FuSeBMC_LICENSE.txt Add files via upload 6 months ago
Makefile Update Makefile 4 months ago
About
FuSeBMC is a White-Box Fuzzer that combines FUzzing with Symbolic Execution via Bounded Model Checking to verify intricate properties in real-world C programs.
Readme
MIT License
Releases 3
FuSeBMC v.3.6.6 Latest
on Dec 20, 2020
+ 2 releases
Packages
No packages published
Publish your first package
Languages
Python 45.9%
C++ 35.4%

Thank you…

Questions?!