Automated Compatibility Testing Method for Software Logic by Using Symbolic Execution

Keiji Uetsuki
FeliCa Networks, Inc.

InSTA 2015
2nd International Workshop on Software Test Architecture

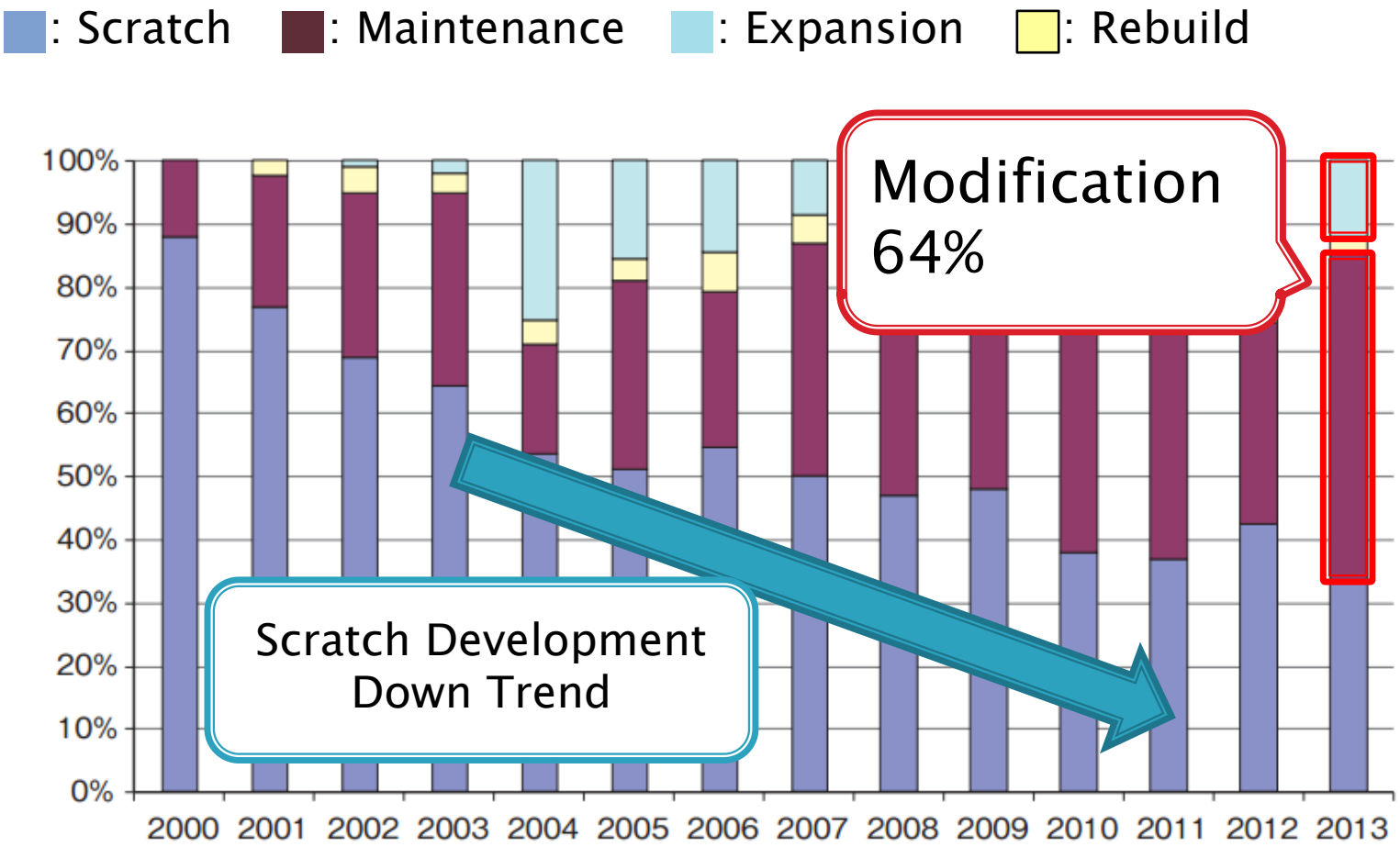
Agenda

- Background
- Current Compatibility Testing Process
- Proposed Testing Process
- Application Experiment
- Discussion to apply to commercial software
- Conclusion

Background

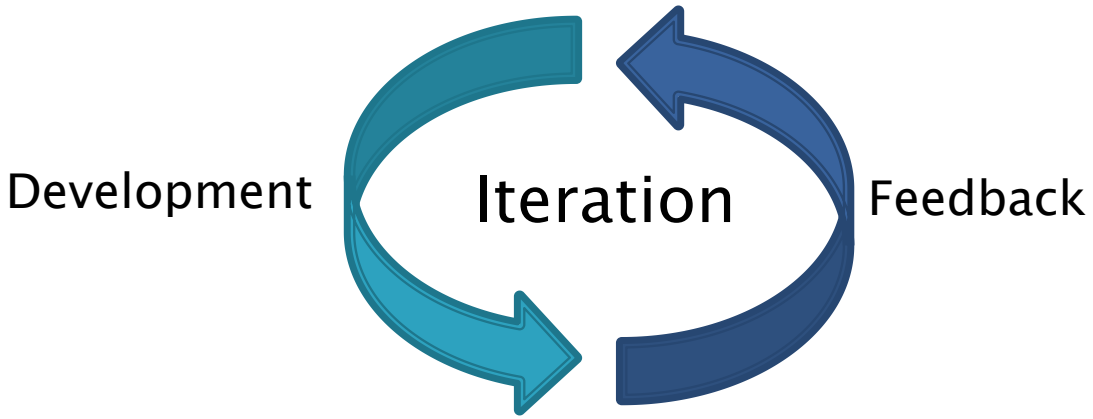
- Software is everywhere in today's world
- Context where software is working is changing because world (our society) is changing
- Software must be updated to keep matching context with our real world
- Longer the software lifecycle, more changing demand

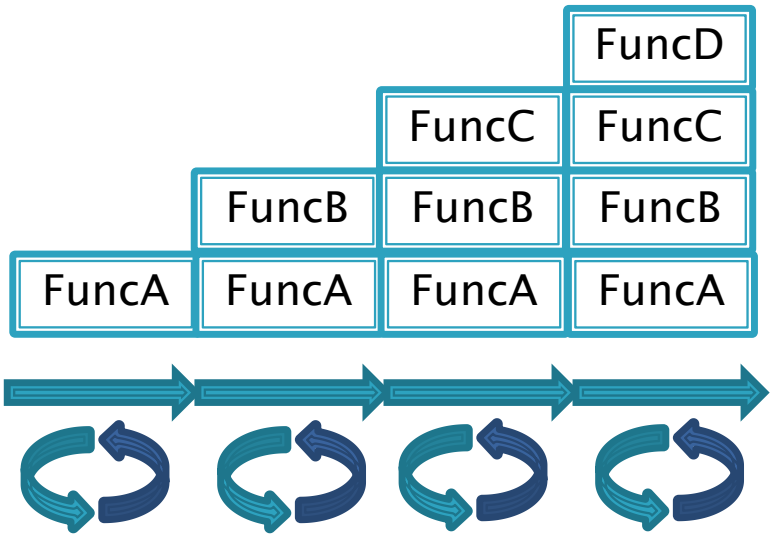
Trend of Software Development



Source: IPA/SEC White Paper 2014-2015 on Software Development Projects in Japan

Agile software development process





Issues on Software modification

- Short time of development
  - Modification tends to be treated as easier than scratch
- Lack of knowledge
  - No document, no person in charge
- Compatibility assurance

Our interest

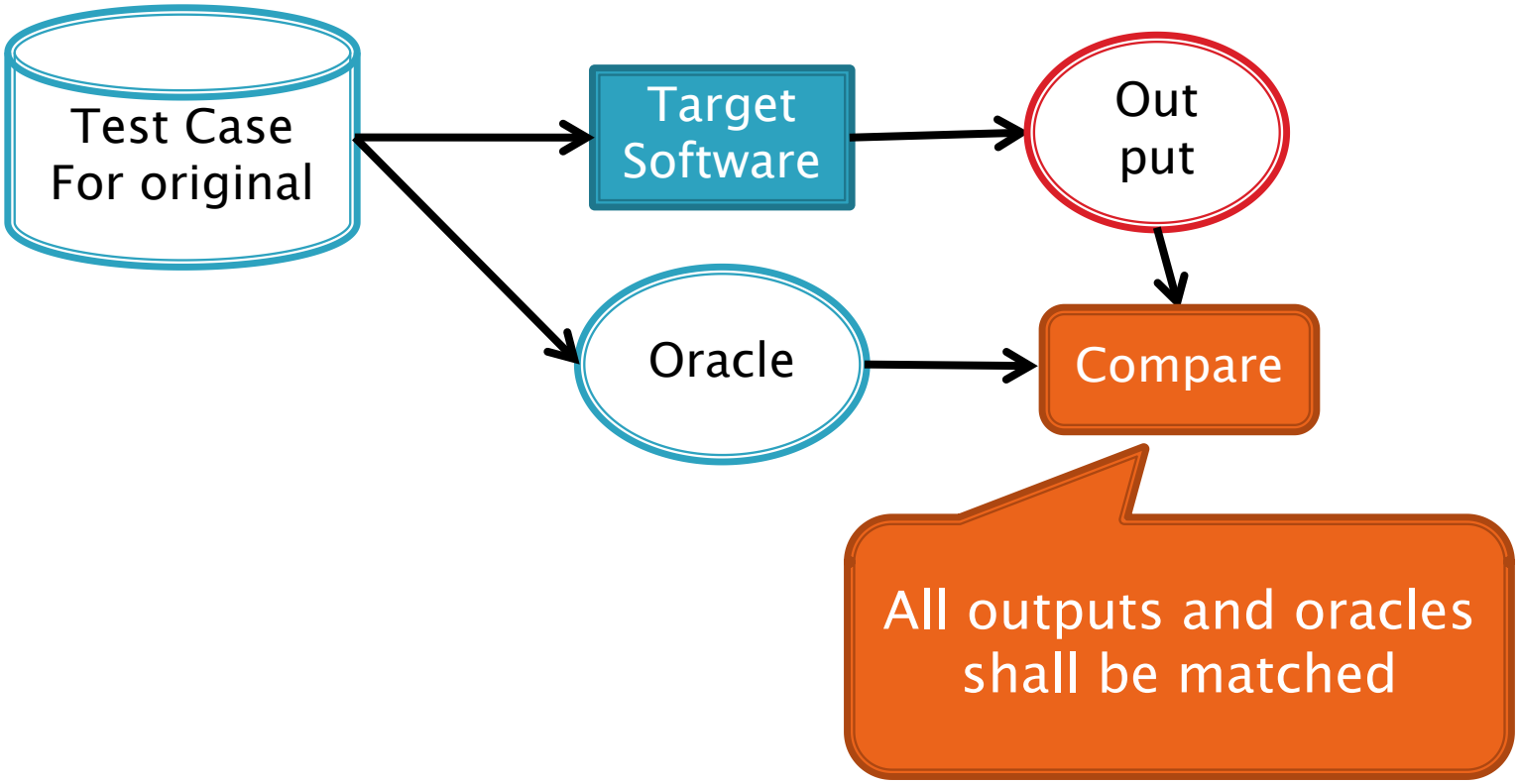
Objective of our study

- Resolve compatibility testing issues by applying Symbolic Execution technique to the testing process

- Scope
  - Software logical behavior
    - there are also some more perspectives such as performance but out of scope
  - Software modification does not change the user interface
    - Test cases of the original software can be re-used

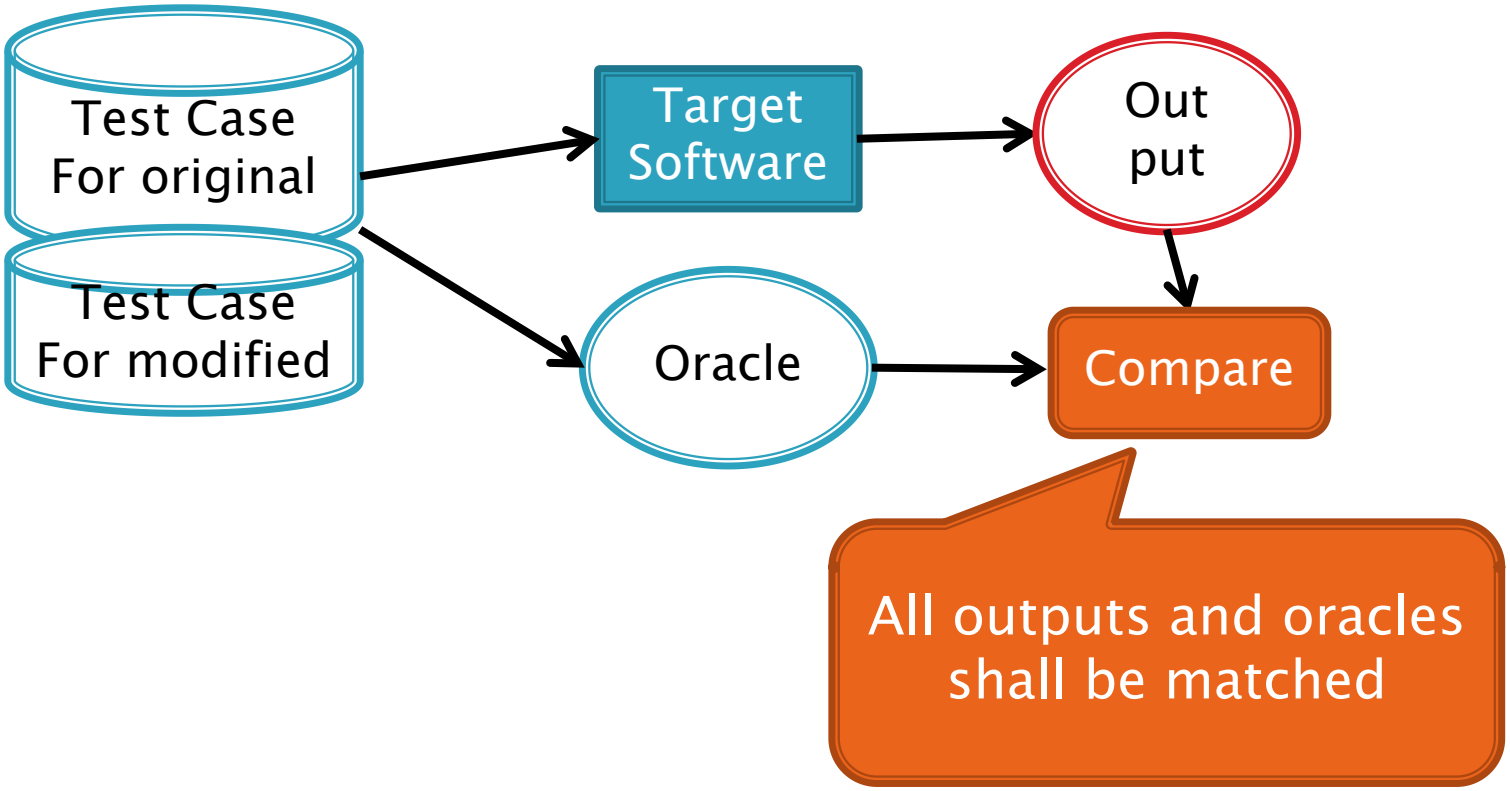
Compatibility testing process

<u>Testing process for original software</u>



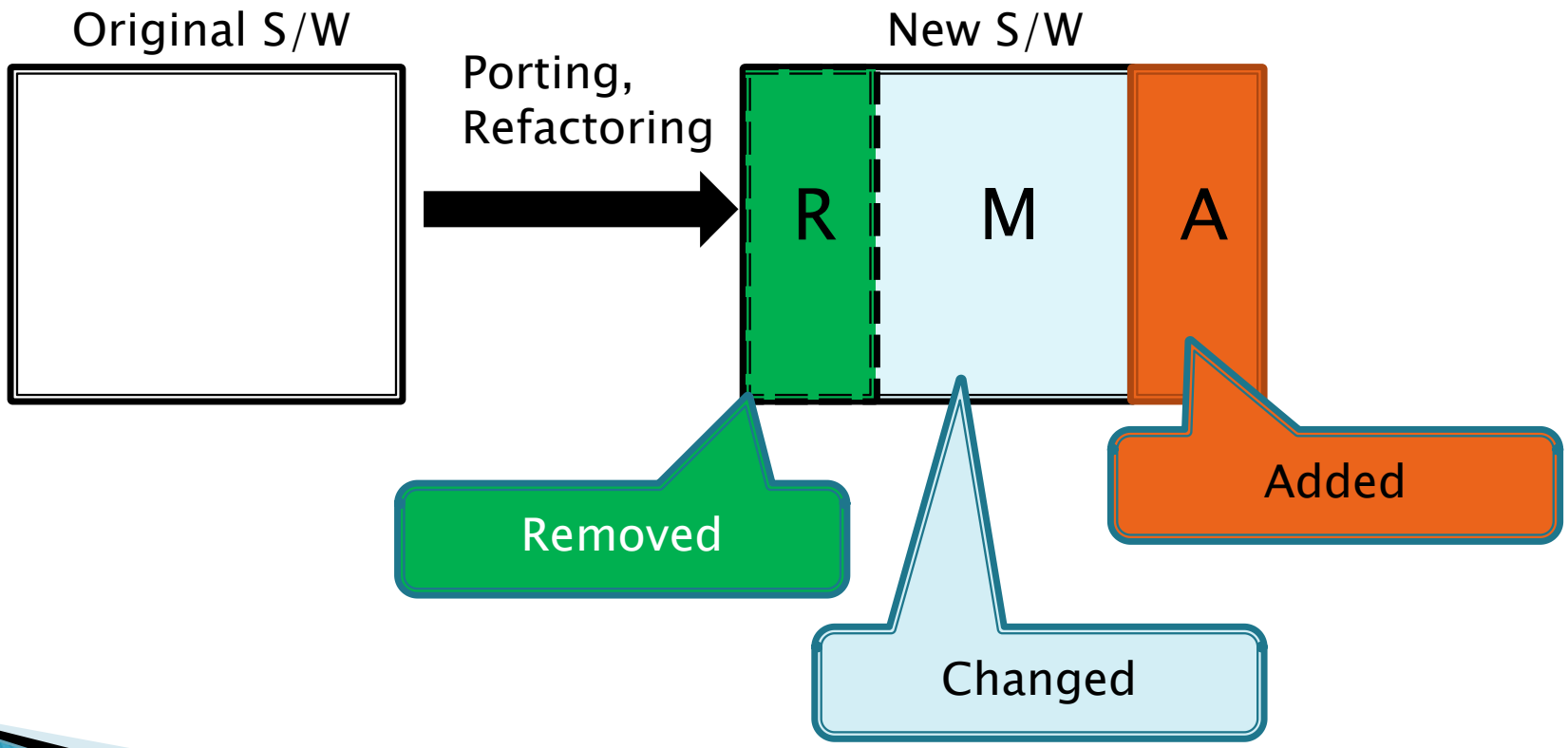
Compatibility testing process

<u>Testing process for modified software</u>



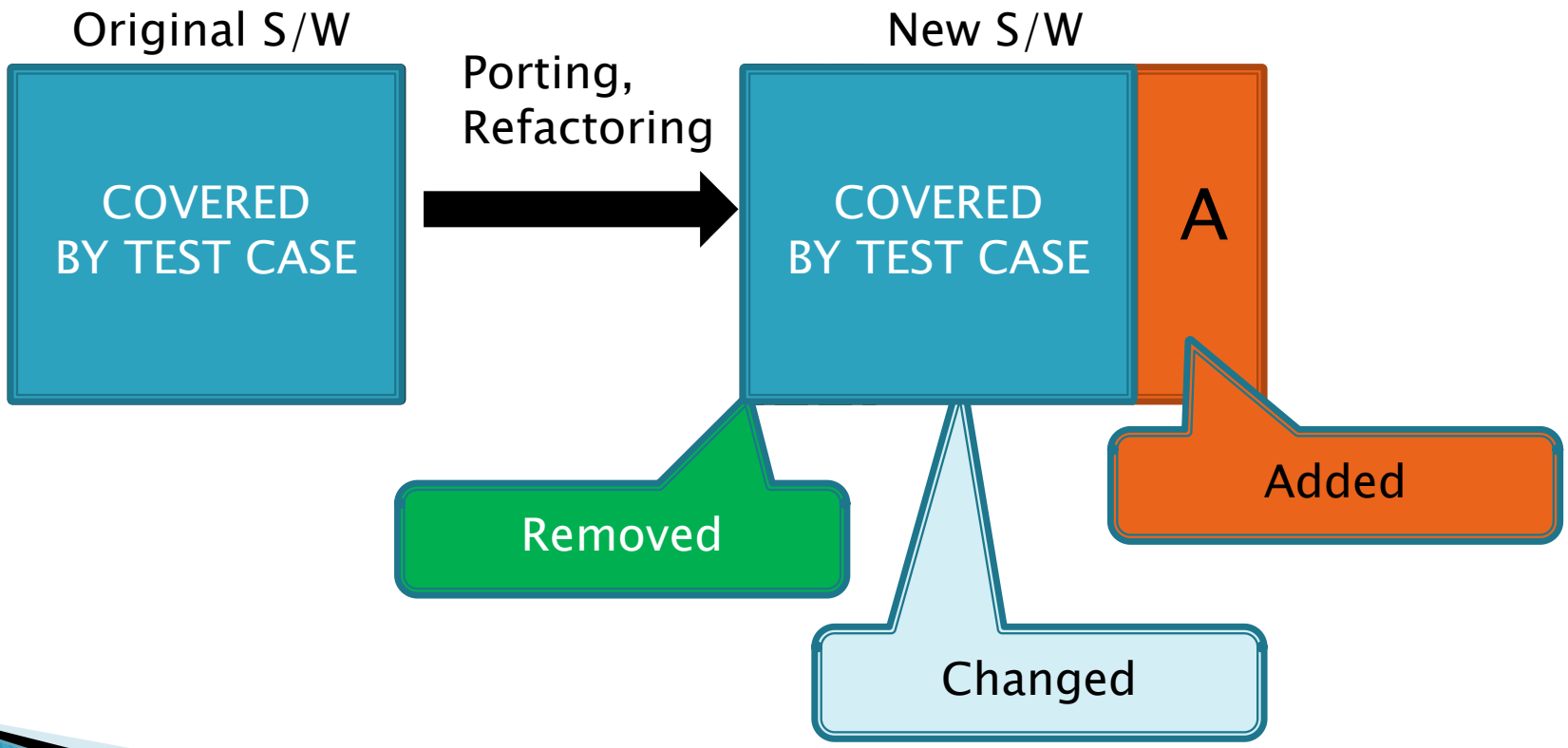
Test coverage

Can we find all bugs ?



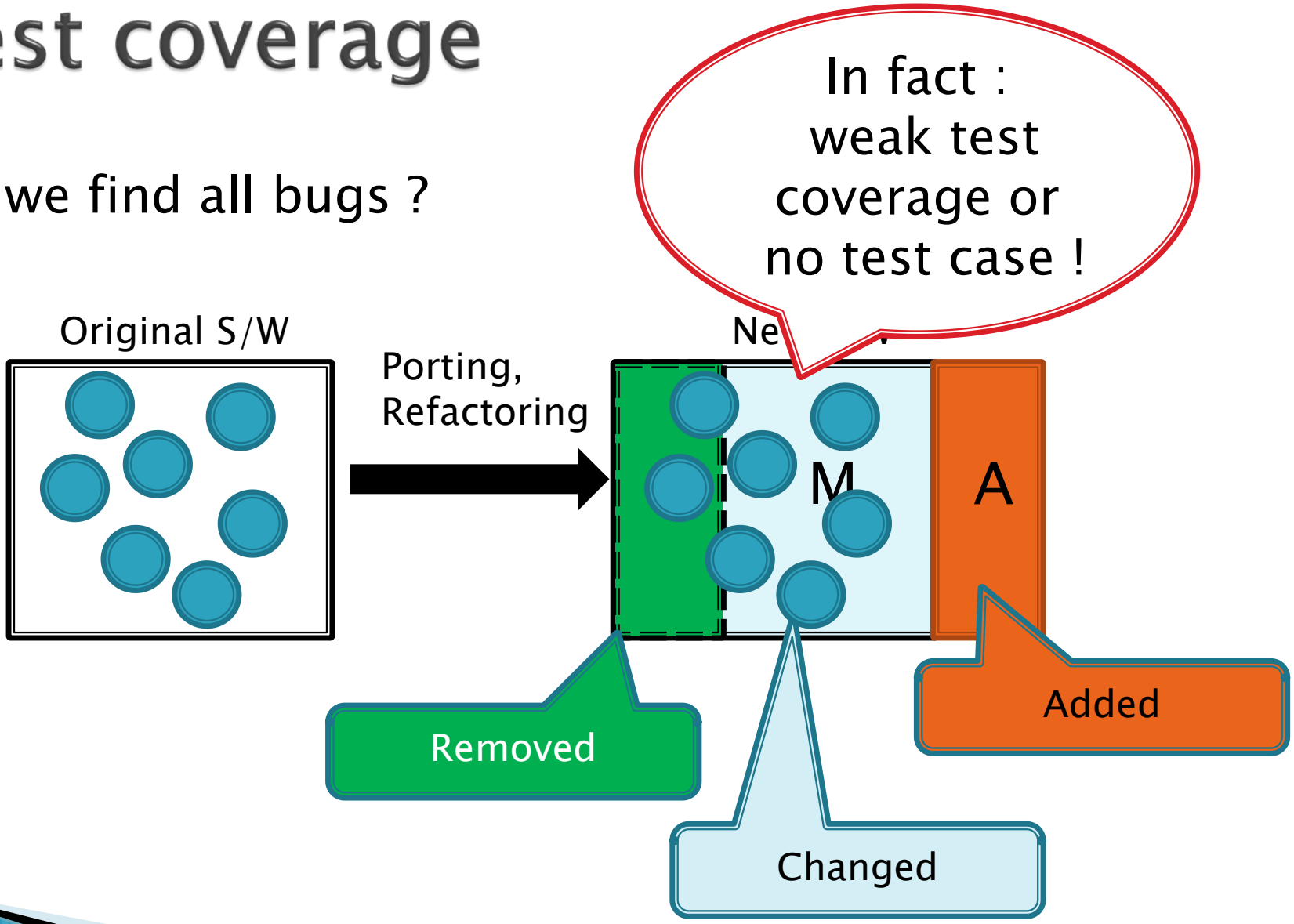
Test coverage

Can we find all bugs ?

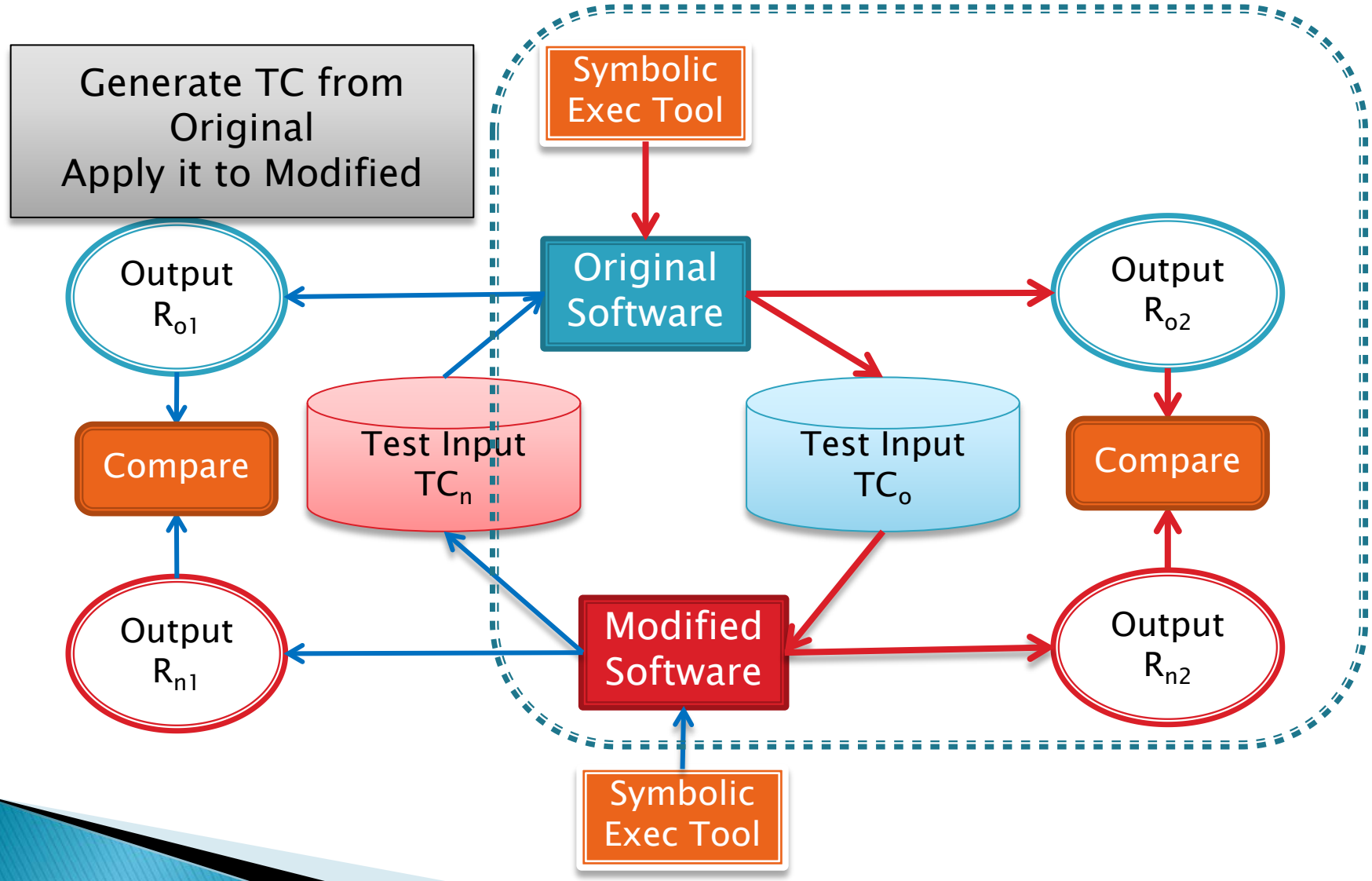


Test coverage

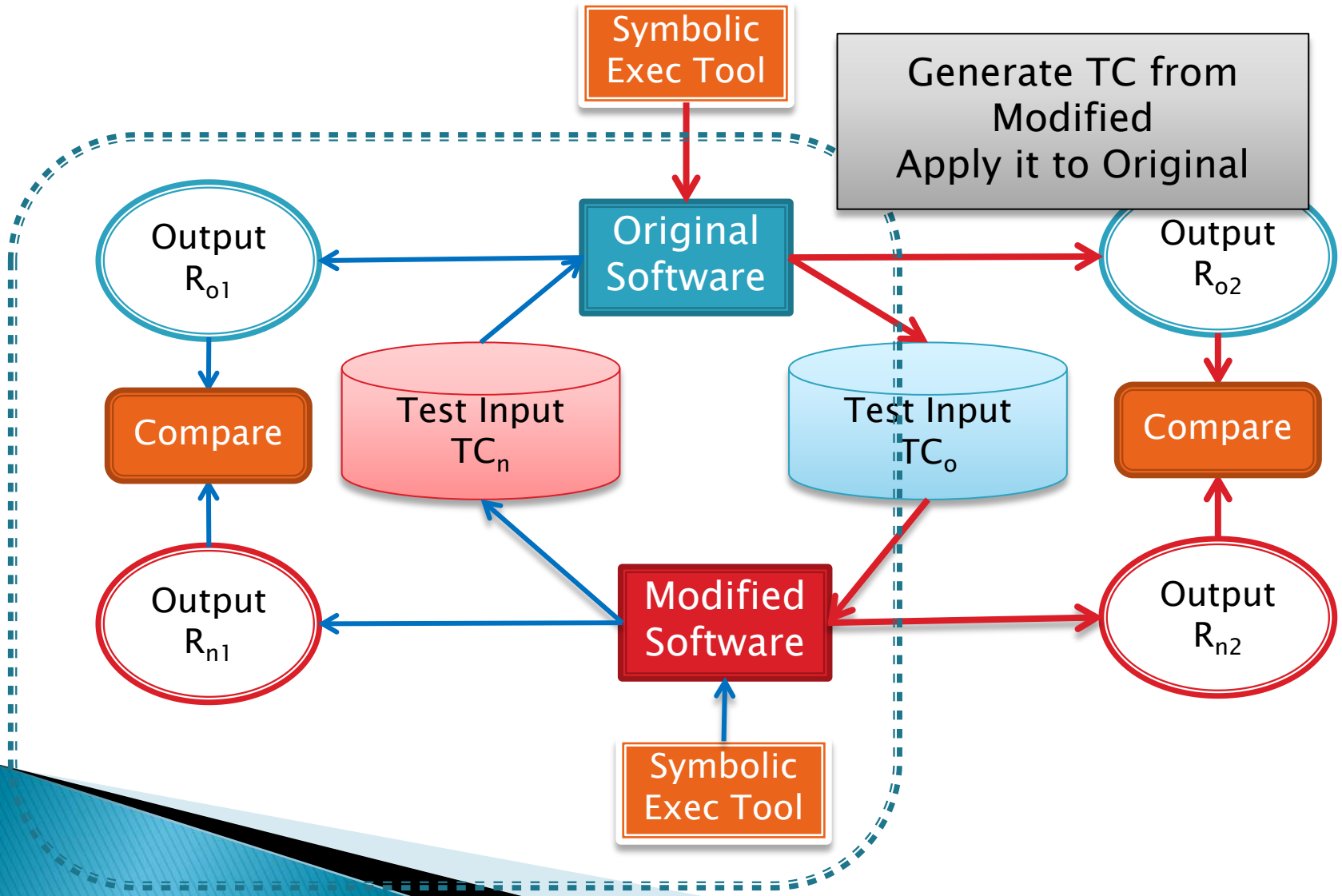
Can we find all bugs ?



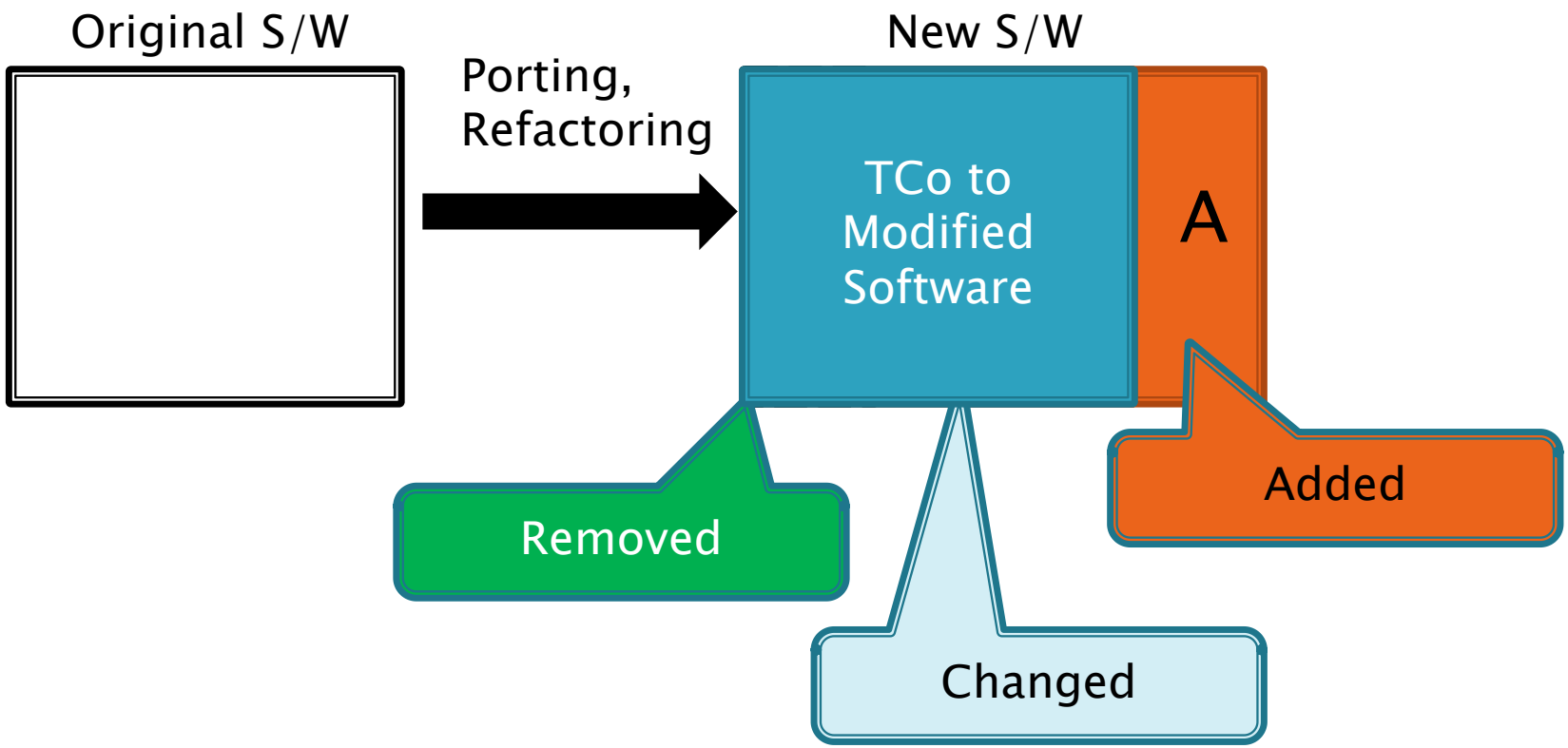
Proposed testing process



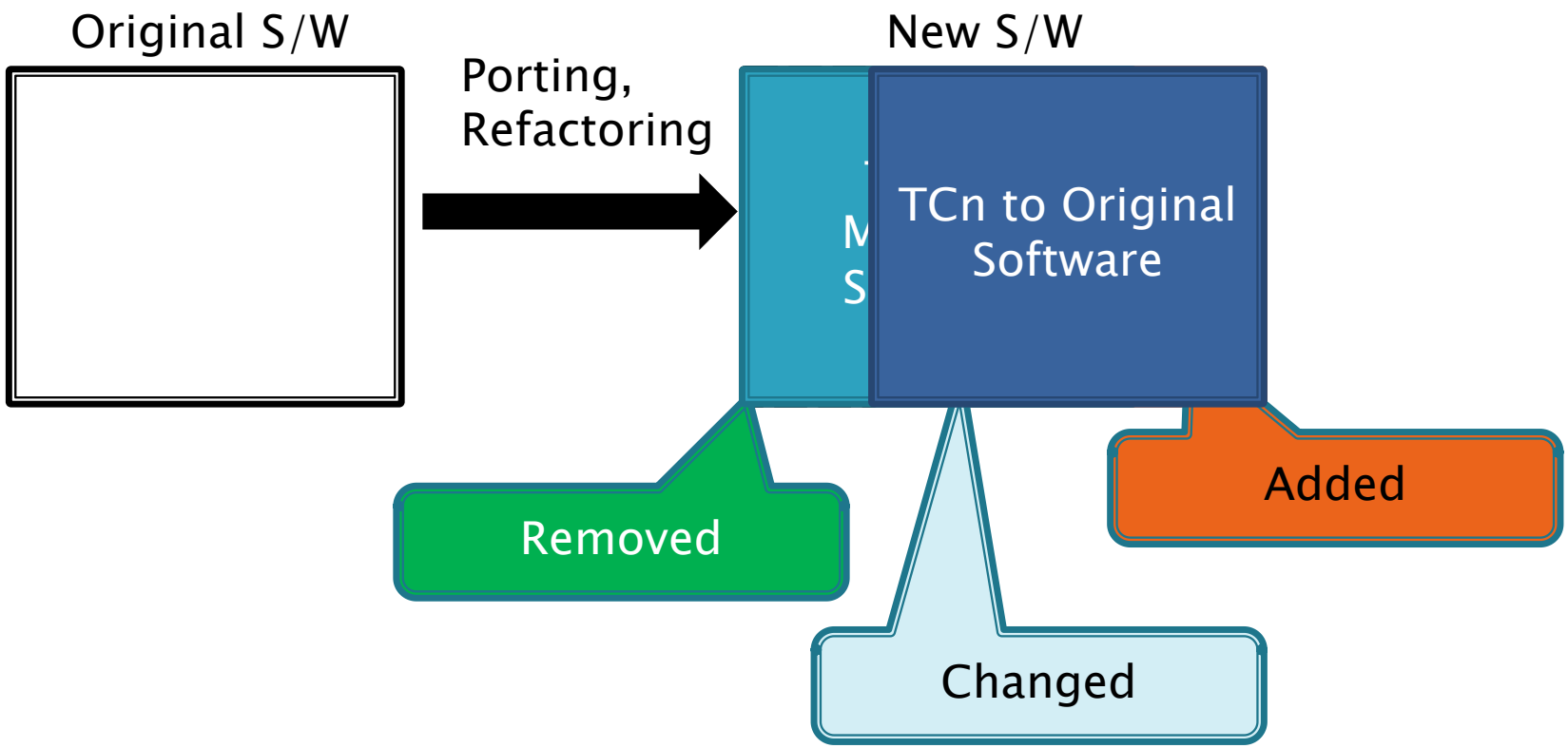
Proposed testing process



Test Coverage



Test Coverage



Application Experiment

- Experiment 1
  - Apply this method to the sample application in case of specification modification
  - To verify if we can find three types of modification: Add, Change, Remove
- Experiment 2
  - Apply this method to the sample application in case that the application has bugs
  - To verify if we can find three types of bugs

Specification of the application

Sample application calculates a discount rate of admission fee for the public facility

- If age of a customer is less than or equal to 3, Free (0% of the normal fee)
- If Wednesday, 90% of the normal fee
- If age of a customer is greater than or equal to 60, 60% of the normal fee
- If sex of a customer is female and her age is greater than or equal to 50, 65% of the normal fee
- If the memorial day, 80% of the normal fee
- If a customer is local citizen, 50% of the normal fee
- If after 3 P.M., 70% of the normal fee
- If age of a customer is less than or equal to 12, 40% of the normal fee
- Note that greater discount rate is applied when multiple conditions are met

Modified specification

- If age of a customer is less than or equal to 3, Free (0% of the normal fee)
- If Wednesday, 90% of the normal fee
- If age of [illegible] is greater than or equal to 60, 60% of the normal [illegible]
- If sex of a customer is female and her age is greater than or equal to **55**, 65% of the normal fee
- ~~If the memorial day, 80% of the normal fee~~
- If a customer is local citizen, 50% of the normal fe[illegible]
- If after 3 P.M., 70% of the normal fee
- If age of a customer is less than or equal to 12, 40% of the normal fee
- **If January or February, 67% of the normal fee**
- Note that greater discount rate is applied when [illegible] conditions are met

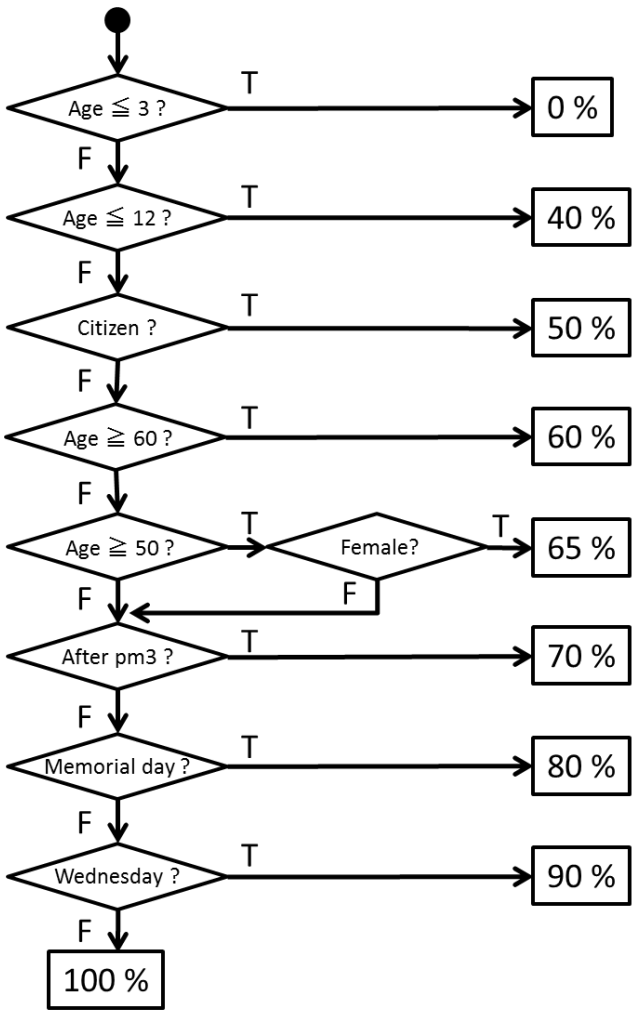
Change

Remove

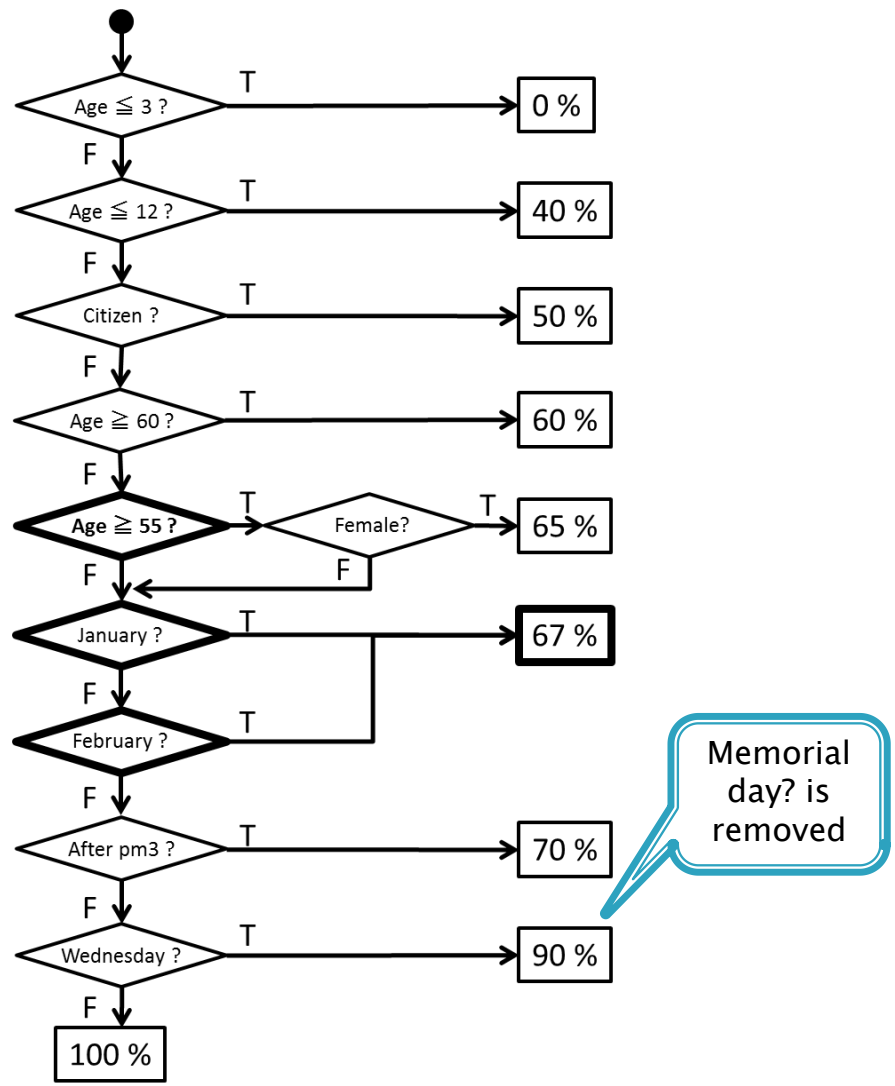
Add

Flow chart of the program

Original Specification



Modified Specification



Tool demonstration

- Symbolic Execution Tool for Java : SPF(Symbolic Path Finder)

- See here for detail of SPF:
  - http://babelfish.arc.nasa.gov/trac/jpf/wiki/projects/jpf-symbc/doc

Test result from the original specification

#	Sex	Age	Day of week	local citizen?	Month	memorial day?	Time (hour)	Output (discount rate %)	Output from Modified Spec.
1	Male	0	M	N	D	No	0	0	0
2	Male	4	M				0	40	40
3	Male	13	M				0	50	50
4	Male	60	M				0	60	60
5	Female	50	M				0	65	**100**
6	Female	13	Monday	No	December	No	15	70	70
7	Female	13	M				0	80	**100**
8	Female	13	W				0	90	90
9	Female	13	M				0	100	100
10	Male	13	M				15	70	70
11	Male	13	Monday	No	December	Yes	0	80	**100**
12	Male	13	Wednesday	No	December	No	0	90	90
13	Male	13	Monday	No	December	No	0	100	100

Because the condition for age of customer is changed

Because the condition "Memorial day?" is removed

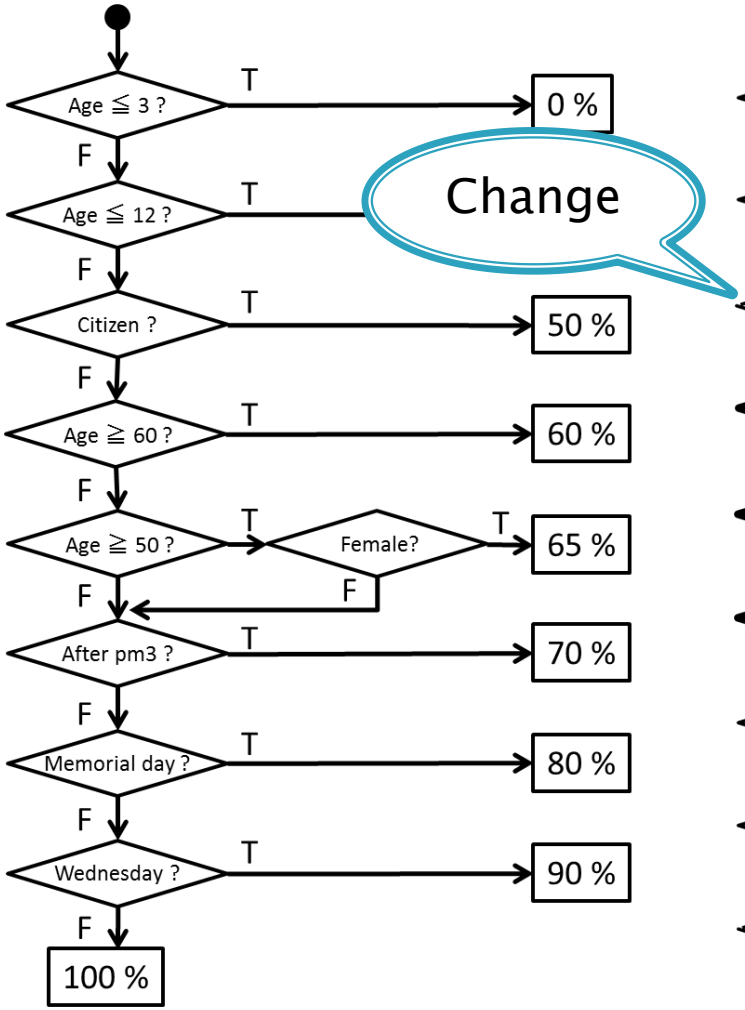
Test result from the modified specification

#	Sex	Age	Day of week	local citizen?	Month	memorial day?	Time (hour)	Output (discount rate %)	Output from Original Spec.
1	Male	0	Monday	No	December	No	0	0	0
2	Male	4	Monday	No	December	No	0	40	40
3	Male	13	Monday	Yes	December	No	0	50	50
4	Male	60	Monday	No	December	No	0	60	60
5	Female	55	Monday	No	December	No	0	65	65
6	Female	13	Monday	No	February	No	15	67	**100**
7	Female	13	Monday	No	December	No	15	70	70
8	Female	13	Wed				0	90	90
9	Female	13	M				0	100	100
10	Female	13	M				0	67	**100**
11	Male	13	M				0	67	**100**
12	Male	13	Mo			o	15	70	70
13	Male	13	Wednesday	No	December	No	0	90	90
14	Male	13	Monday	No	December	No	0	100	100
15	Male	13	Monday	No	January	No	0	67	**100**

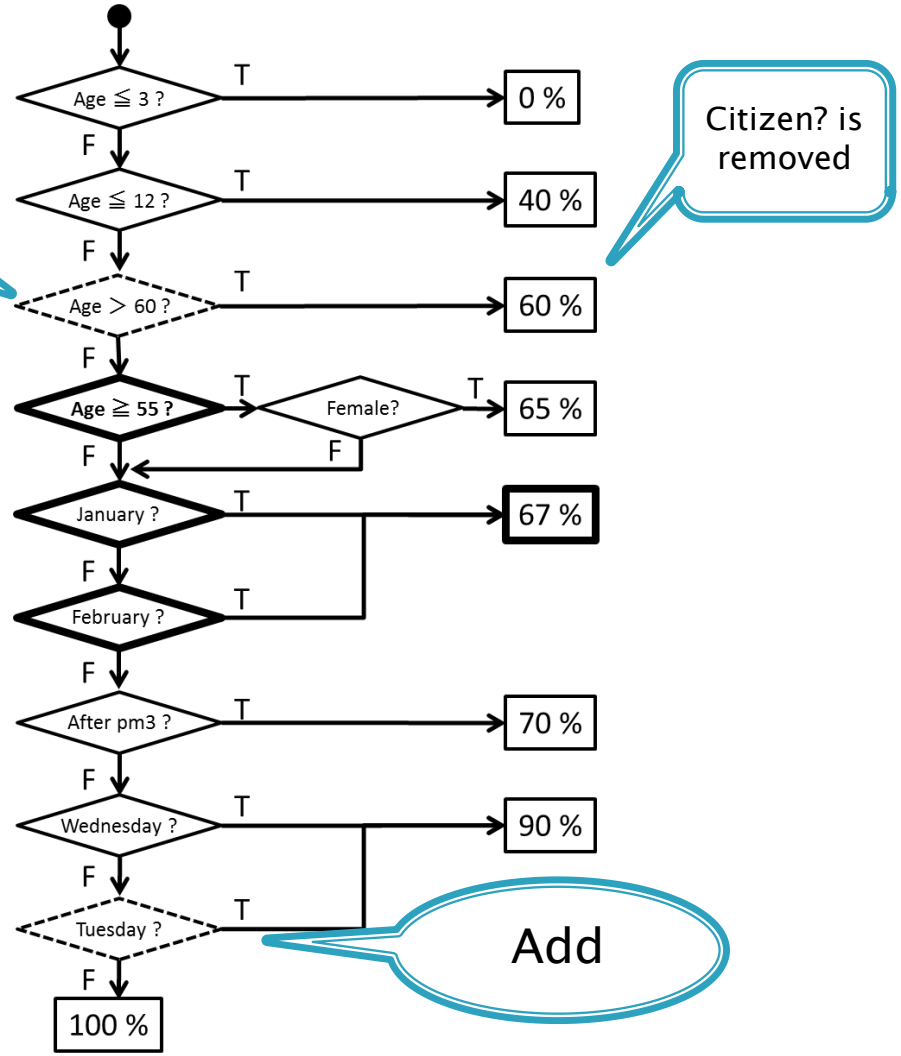
Because the condition for Month is newly added

Bug implementation

Original Specification



Bug implemented program



Test result from the original program

#	Sex	Age	Day of week	local citizen?	Month	memorial day?	Time (hour)	Output (discount rate %)	Output from Bug impl.
1	Male	0	[illegible]	[illegible]	[illegible]	No	0	0	0
2	Male	4	[illegible]	[illegible]	[illegible]	No	0	40	40
3	Male	13	[illegible]	[illegible]	[illegible]	[illegible]	0	50	**100**
4	Male	60	[illegible]	[illegible]	[illegible]	No	0	60	**100**
5	Female	50	Monday	No	December	No	0	65	**100**
6	Female	13	[illegible]	[illegible]	[illegible]	[illegible]	15	70	70
7	Female	13	[illegible]	[illegible]	[illegible]	Yes	0	80	**100**
8	Female	13	[illegible]	[illegible]	[illegible]	No	0	90	90
9	Female	13	[illegible]	[illegible]	[illegible]	No	0	100	100
10	Male	13	Monday	No	December	No	15	70	70
11	Male	13	Monday	No	December	Yes	0	80	**100**
12	Male	13	Wednesday	No	December	No	0	90	90
13	Male	13	Monday	No	December	No	0	100	100

Because the condition "Citizen?" is removed

Because inequality sign of the condition for age is changed

Test result from the bug implemented program

#	Sex	Age	Day of week	local citizen?	Month	memorial day?	Time (hour)	Output (discount rate %)	Output from Original Spec.
1	Male	0	Monday	No	December	No	0	0	0
2	Male	4	Monday	No	December	No	0	40	40
3	Male	61	Monday	No	December	No	0	60	60
4	Female	55	Monday	No	December	No	0	65	65
5	Female	13	Monday	No	February	No	0	67	**100**
6	Female	13	Monday	No	December	No	15	70	70
7	Female	13	Wednesday	No	December	No	0	90	90
8	Female	13	Tuesday	No	December	No	0	90	**100**
9	Female	13	Monday	No	December	No	0	100	100
10	Female	13	Mo[illegible]	[illegible]	[illegible]	No	[illegible]	67	**100**
11	Male	13	M[illegible]	[illegible]	[illegible]	[illegible]	0	67	**100**
12	Male	13	M[illegible]	[illegible]	[illegible]	[illegible]	15	70	70
13	Male	13	We[illegible]	[illegible]	[illegible]	[illegible]	0	90	90
14	Male	13	T[illegible]	[illegible]	[illegible]	[illegible]	0	90	**100**
15	Male	13	M[illegible]	[illegible]	[illegible]	[illegible]	0	100	100
16	Male	13	Monday	No	January	No	0	67	**100**

Because the condition "Tuesday?" is newly added

Discussion

- Is there any issue when we apply the method to a commercial software?

Issues to be solved

- Tool dependency
- Scalability
- Education

ISSUE: Tool dependency

- Our method uses Symbolic execution tool to generate test cases

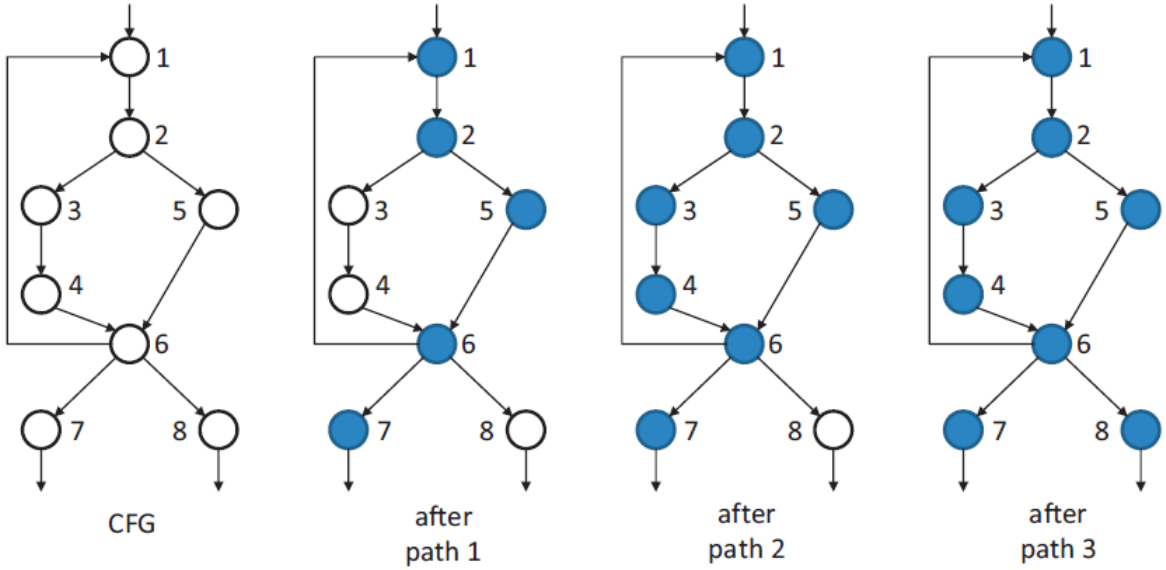
- Some restrictions on the tools
  - Supported Language: C, C++, Java, JavaScript, etc.
  - Supported Input Variables: bit array, integer, floating point, etc.

ISSUE: Scalability

- Sample program is okay, but commercial program is much bigger
- Issues on scalability has been studied
- 1)Path explosion
  - Too many path to be treated
- 2)Checking differences of test cases
  - Too many differences to be checked by hand

Solution for path explosion

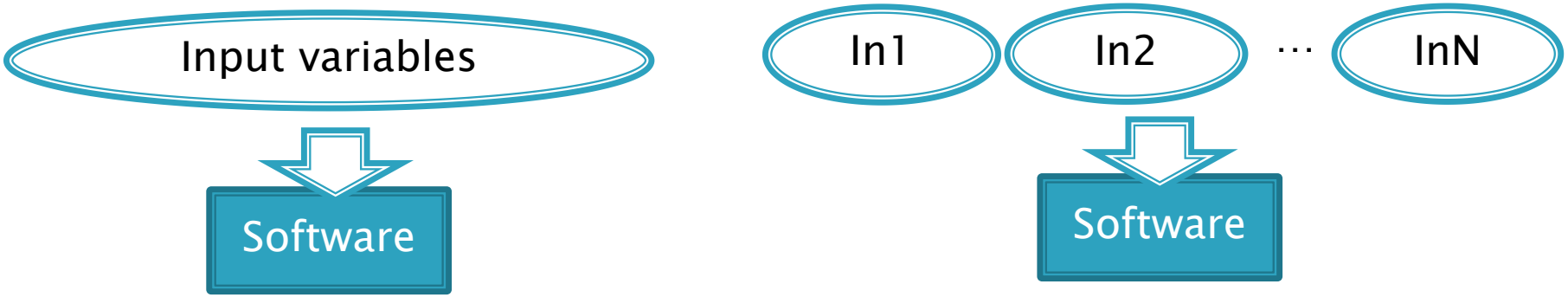
- Path cutting technique
  - Add constraints about coverage level
  - This study introduces condition coverage instead of full path coverage



Source: Enhancing Symbolic Execution to Test the Compatibility of Re-engineered Industrial Software

Solution for path explosion

- Variable grouping
  - If functions of the software can be treated independently, the input variables can be divided



Source: Enhancing Symbolic Execution to Test the Compatibility of Re-engineered Industrial Software

Solution for difference check

- Bigger modification is made, more differences may be made
- Therefore we must stop big-bang testing, instead, frequently testing at small modification.
- Since the testing process is automated, we can do it.

ISSUE: Education

- Symbolic Execution technique and tools are now open
- However it is still not easy to understand and use
- Due to the tool installation, restrictions, no user community in Japan

- Now we are creating
  the user community!

Conclusion

- We proposed a new software testing method for logic compatibility verification.
- By using this method compatibility of logical behavior was exhaustively verified and also full path coverage was achieved.
- From the experimental results, it was verified that the method could detect all the three types of specification changes and bugs.
- Our next step is to apply the method to many types of real software to clarify its limitation or restriction which depends on the types.