COMPSAC 2013
2013.07.22 ~ 07.25
Kyoto Japan

SQAF-DS: A Software Quality Assessment Framework for Dependable Systems

2013.07.25

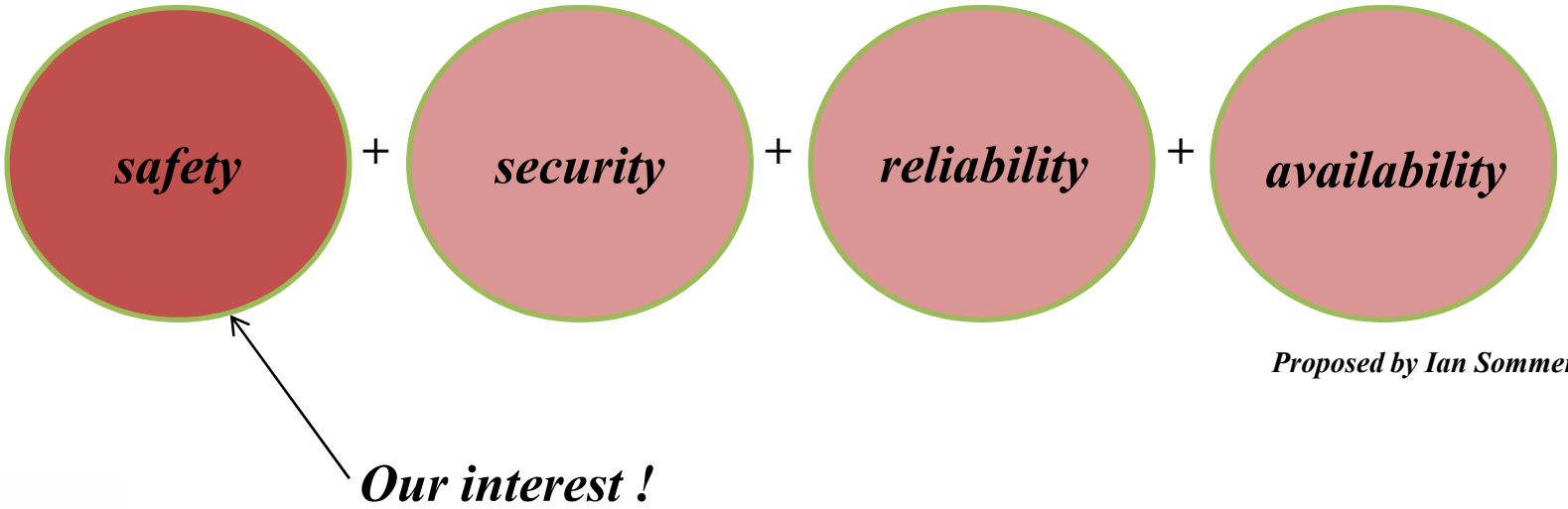
JUNBEOM YOO

KONKUK University
http://dslab.konkuk.ac.kr

Dependability

The extent of the user's confidence that it will operate as they expect and not fail in normal use

A emergent property consisting of

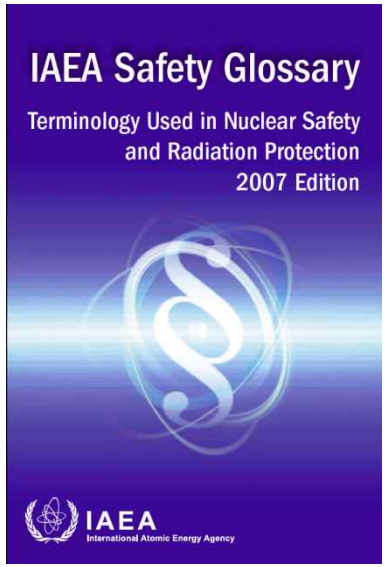


Proposed by Ian Sommerville [41]

“Safety Analysis” of IAEA

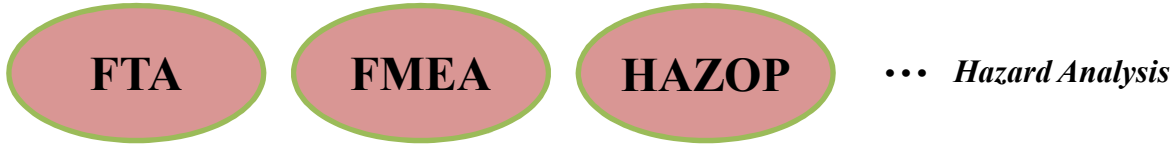
safety analysis. Evaluation of the potential hazards associated with the conduct of an *activity*.

ⓘ *Safety analysis* is often used interchangeably with *safety assessment*. However, when the distinction is important, *safety analysis* should be used for the study of *safety*, and *safety assessment* for the evaluation of *safety* — for example, evaluation of the magnitude of hazards, evaluation of the performance of *safety measures* and judgement of their adequacy, or quantification of the overall radiological impact or *safety* of a *facility* or *activity*.



Safety Analysis Techniques

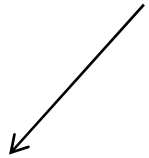
Analysis Techniques **for achieving safety**



Assessment Techniques **for assessing current status of safety**

U.S. NRC. NUREG/CR-6430 [27]

Safety Analysis (Achievement)



" All failures identified by FMEA should be analyzed by FTA, and all potential errors (reasons) identified by FTA should be resolved and confirmed throughout the whole life-cycle of software development."

Safety Assessment

Dependability Assessment

An important activity as well as dependability analysis (achievement)

- **It helps us determine when to stop the analysis effort**

A prompt decision whether to keep the analysis up, while preserving a required level of dependability

- **One of key factors to cost-effective software development**

This Paper Proposes

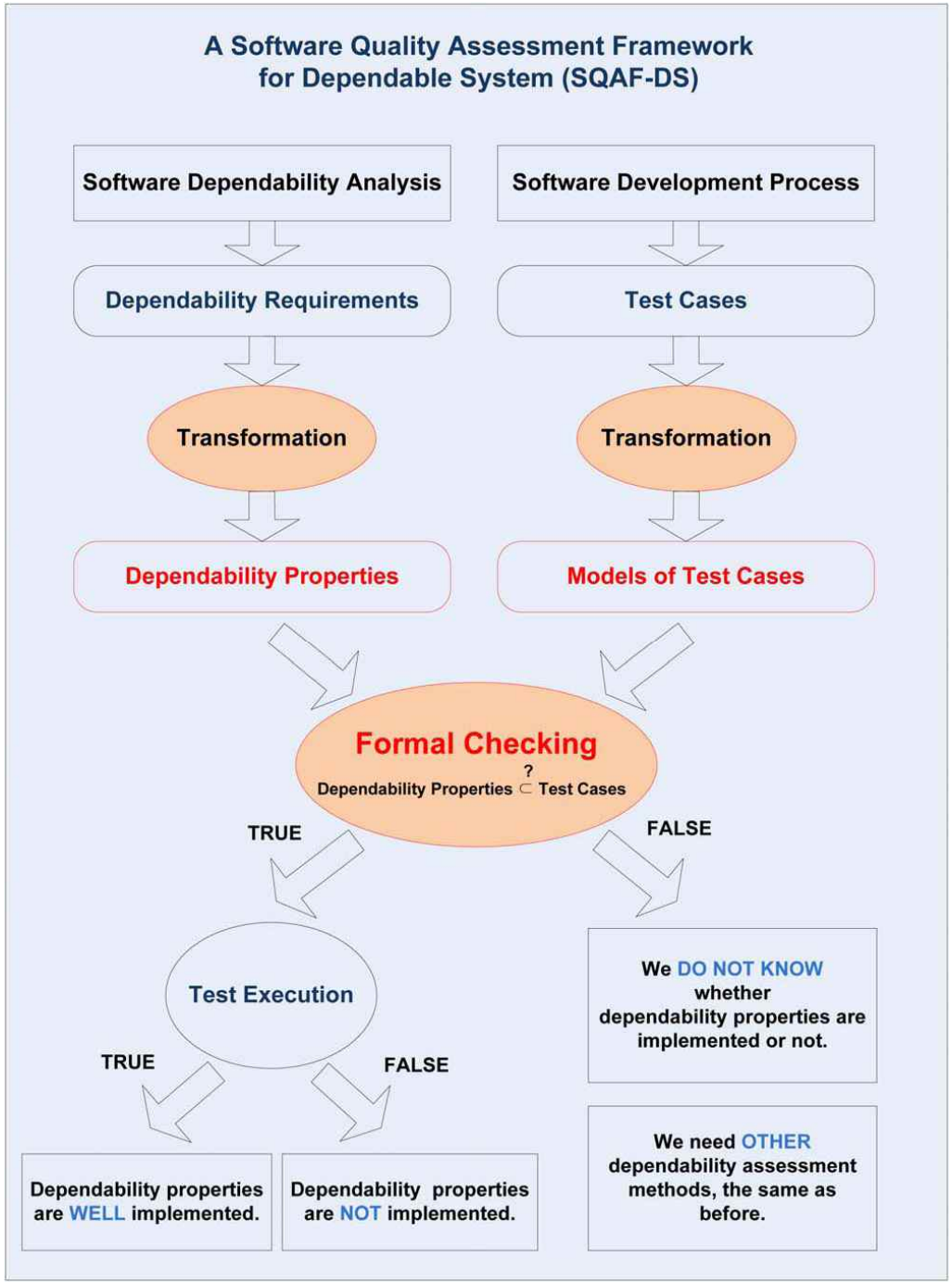
A way to reduce the effort for dependability assessment:

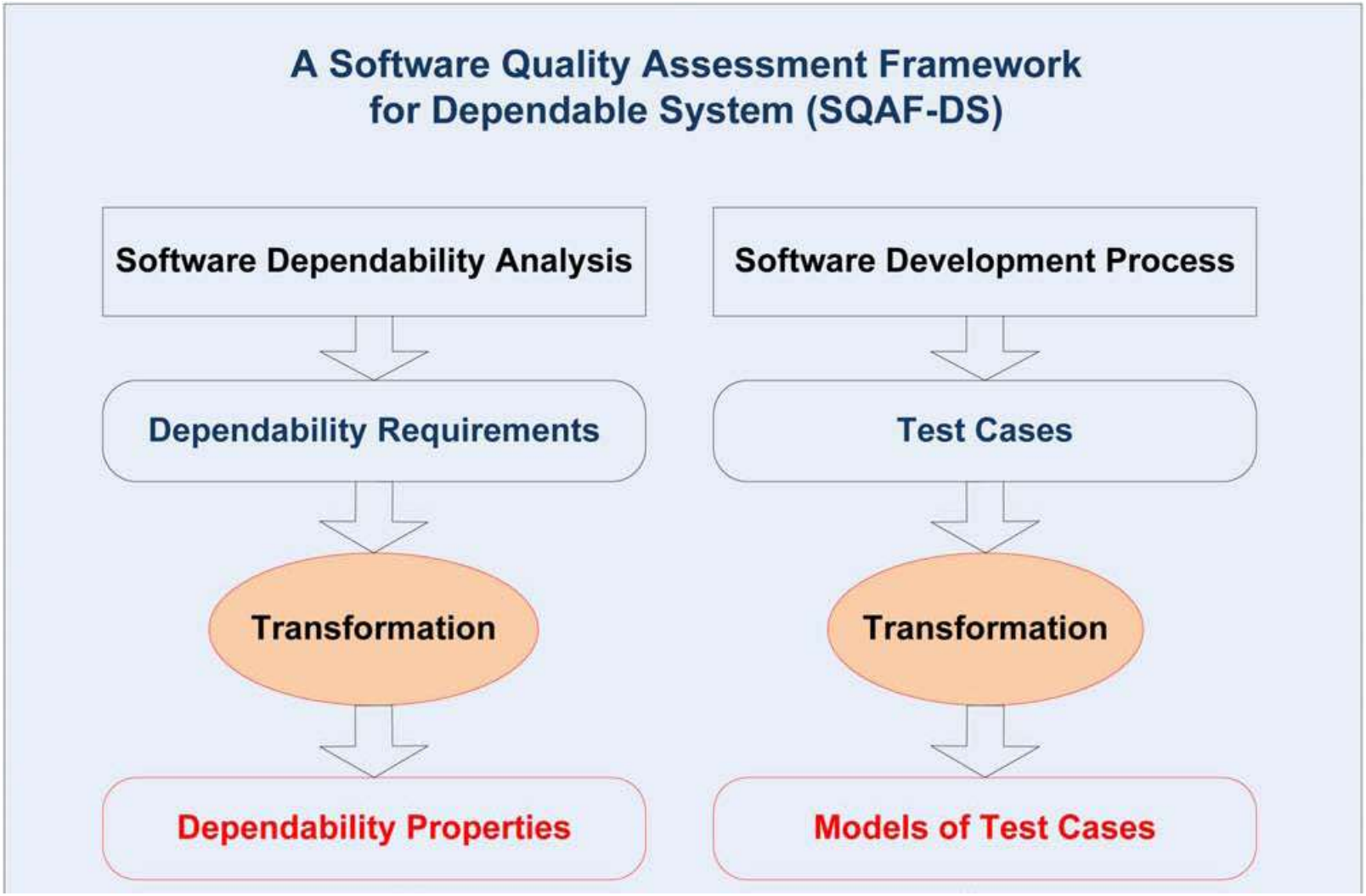
SQAF-DS ***(Software Quality Assessment Framework for Dependable System)***

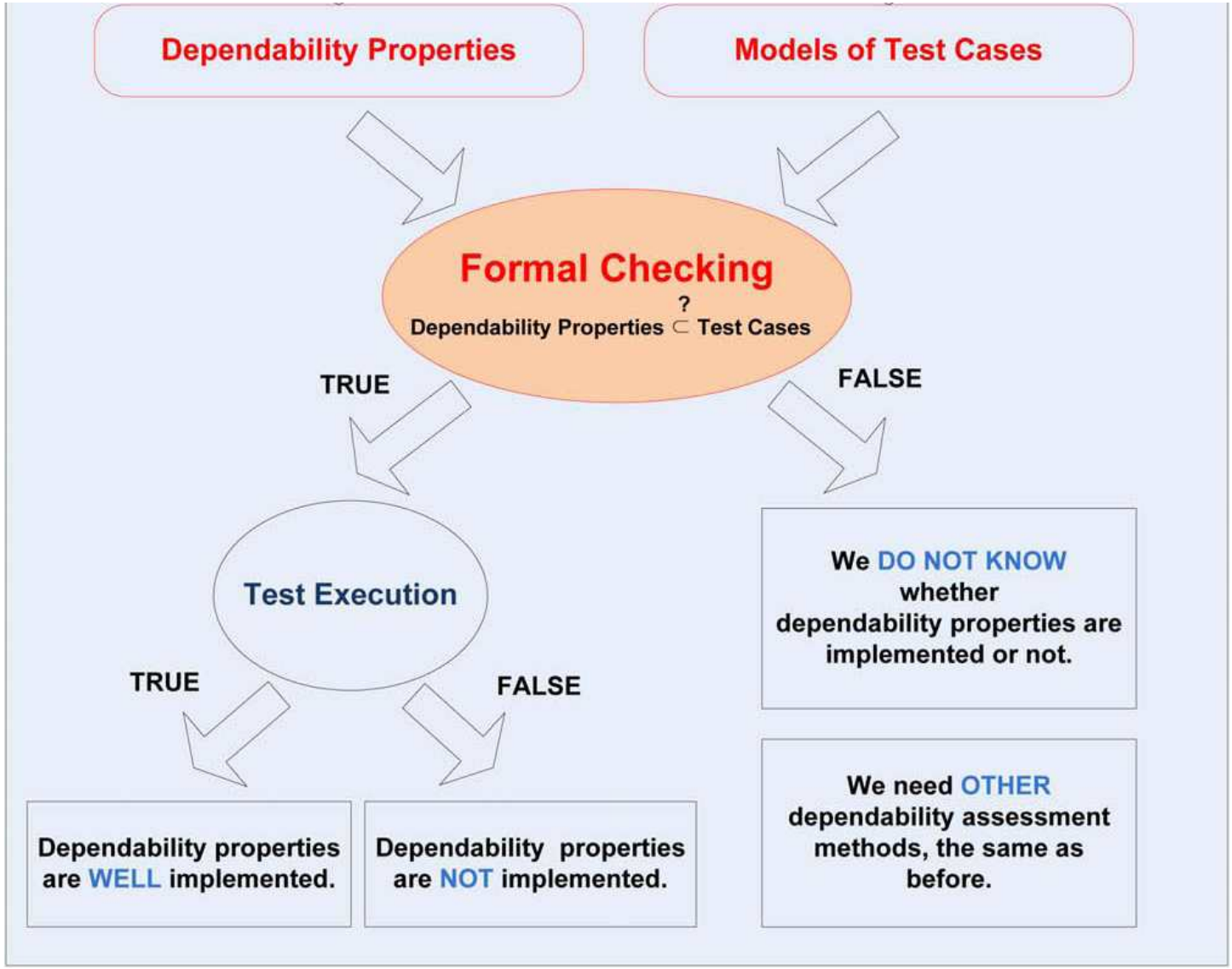
- **Intends to reduce the assessment time and cost thorough using test cases as a means of the assessment**

- **First, develop dependability requirements from dependability analysis**
- **Formally checks inclusion/satisfaction relation between dependability requirements and test cases**

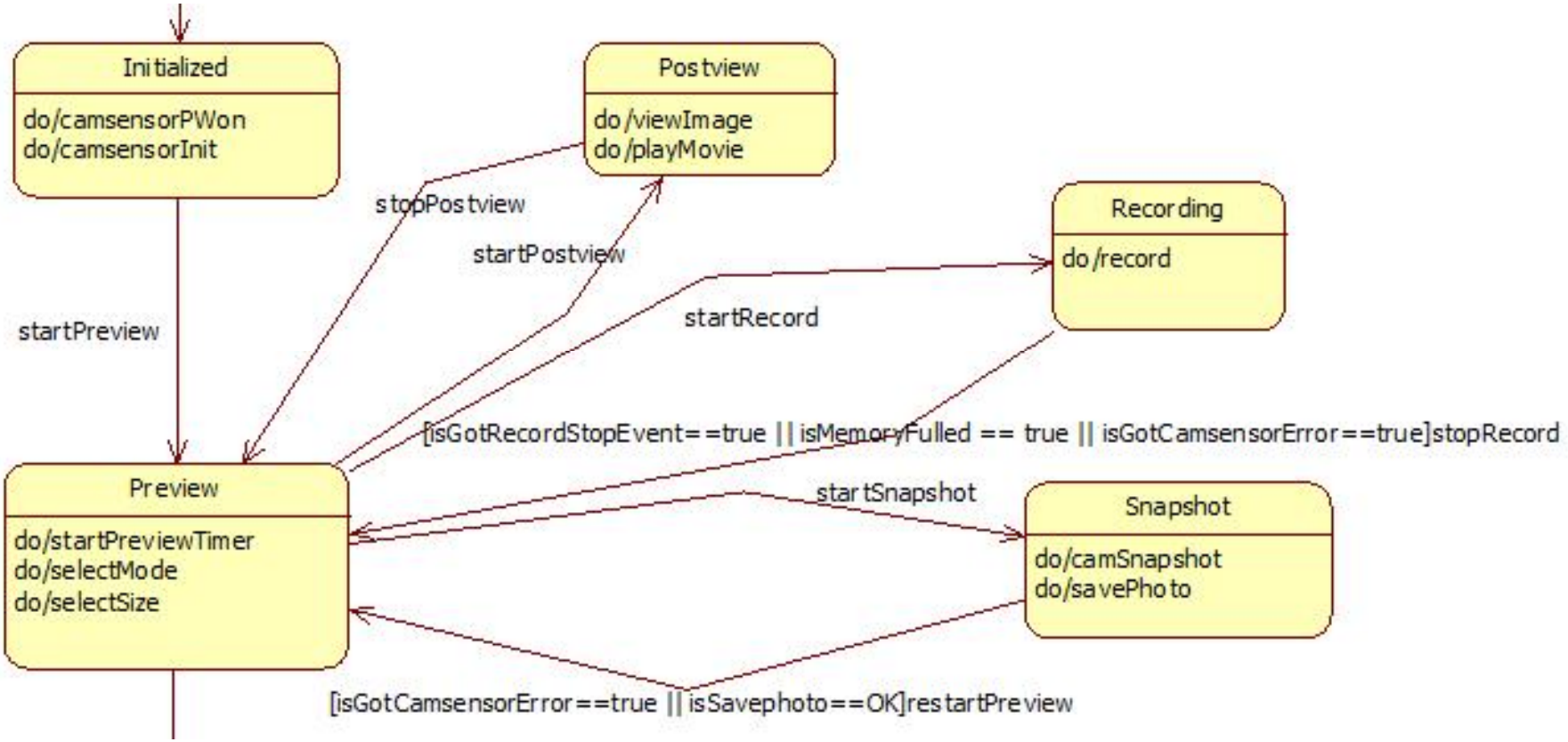
- **Case study: *Safety***


A Software Quality Assessment Framework
for Dependable System (SQAF-DS)
Software Dependability Analysis
Software Development Process
Dependability Requirements
Test Cases
Transformation
Transformation
Dependability Properties
Models of Test Cases
Formal Checking
Dependability Properties ⊂? Test Cases
TRUE
FALSE
Test Execution
We DO NOT KNOW whether dependability properties are implemented or not.
TRUE
FALSE
Dependability properties are WELL implemented.
Dependability properties are NOT implemented.
We need OTHER dependability assessment methods, the same as before.


A Software Quality Assessment Framework
for Dependable System (SQAF-DS)
Software Dependability Analysis
Software Development Process
Dependability Requirements
Test Cases
Transformation
Transformation
Dependability Properties
Models of Test Cases

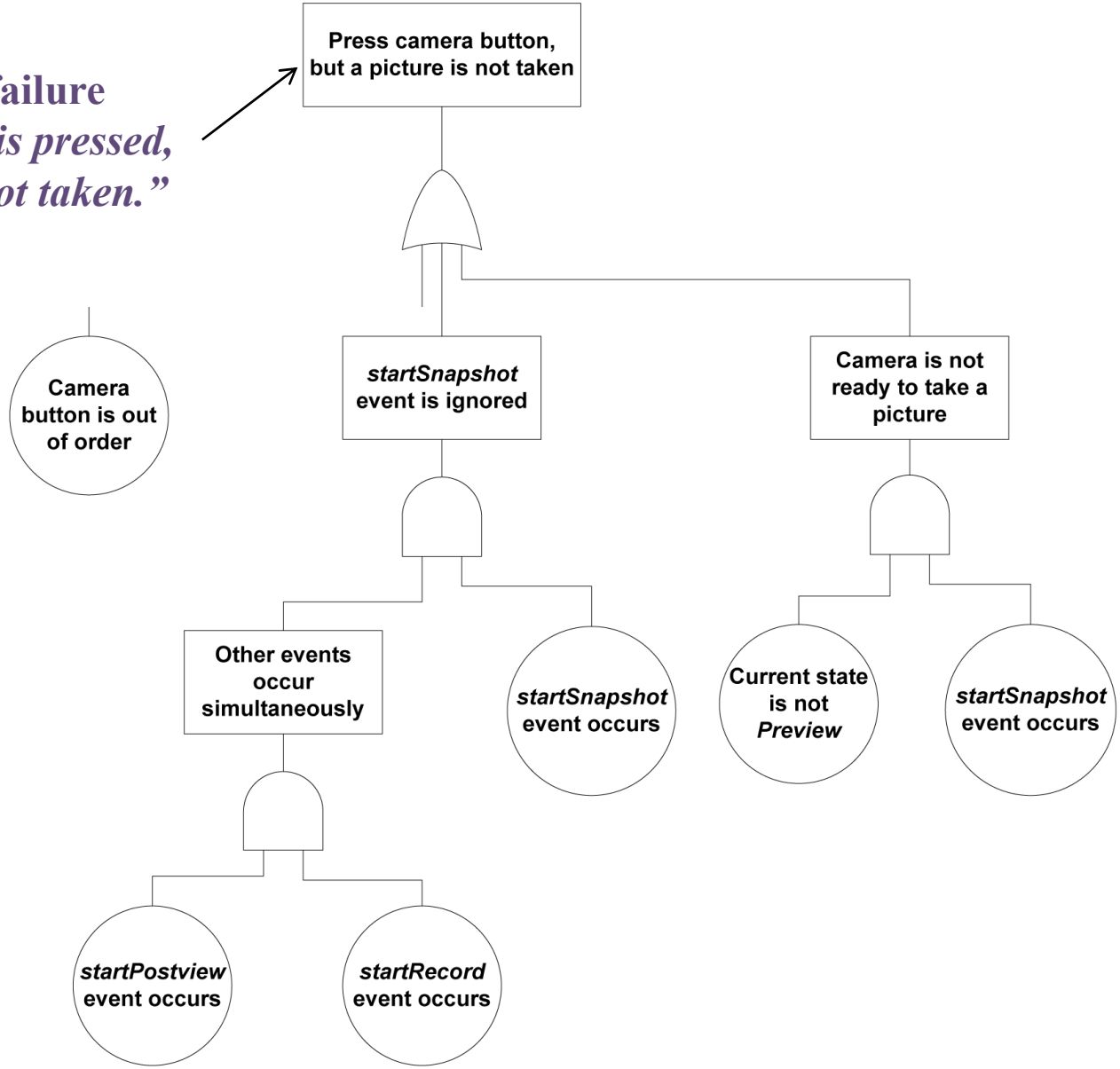

Dependability Properties
Models of Test Cases
Formal Checking
?
Dependability Properties ⊂ Test Cases
TRUE
FALSE
Test Execution
TRUE
FALSE
We DO NOT KNOW whether dependability properties are implemented or not.
We need OTHER dependability assessment methods, the same as before.
Dependability properties are WELL implemented.
Dependability properties are NOT implemented.

Case Study: Camera Control SW of Cell Phones



SW requirement specification in UML (excerpted)

An FTA for the failure *"Camera button is pressed, but a picture is not taken."*



MINIMAL CUT-SET : Camera button is out of order
| startSnapshot event is ignored
| Camera is not ready to take pictures

Safety Requirement (1) :

"If the camera button is pressed, then startSnapshot event should be executed at first, even if three events occur simultaneously."

Safety Requirement (2) :

"If the camera button is pressed when the system is ready to take pictures, then it should take a picture, eventually."

Safety Property (1) :

AG(((state=Preview) & startSnapshot & startRecord & stopPostview) → AX (state=Snapshot))

Safety Property (2) :

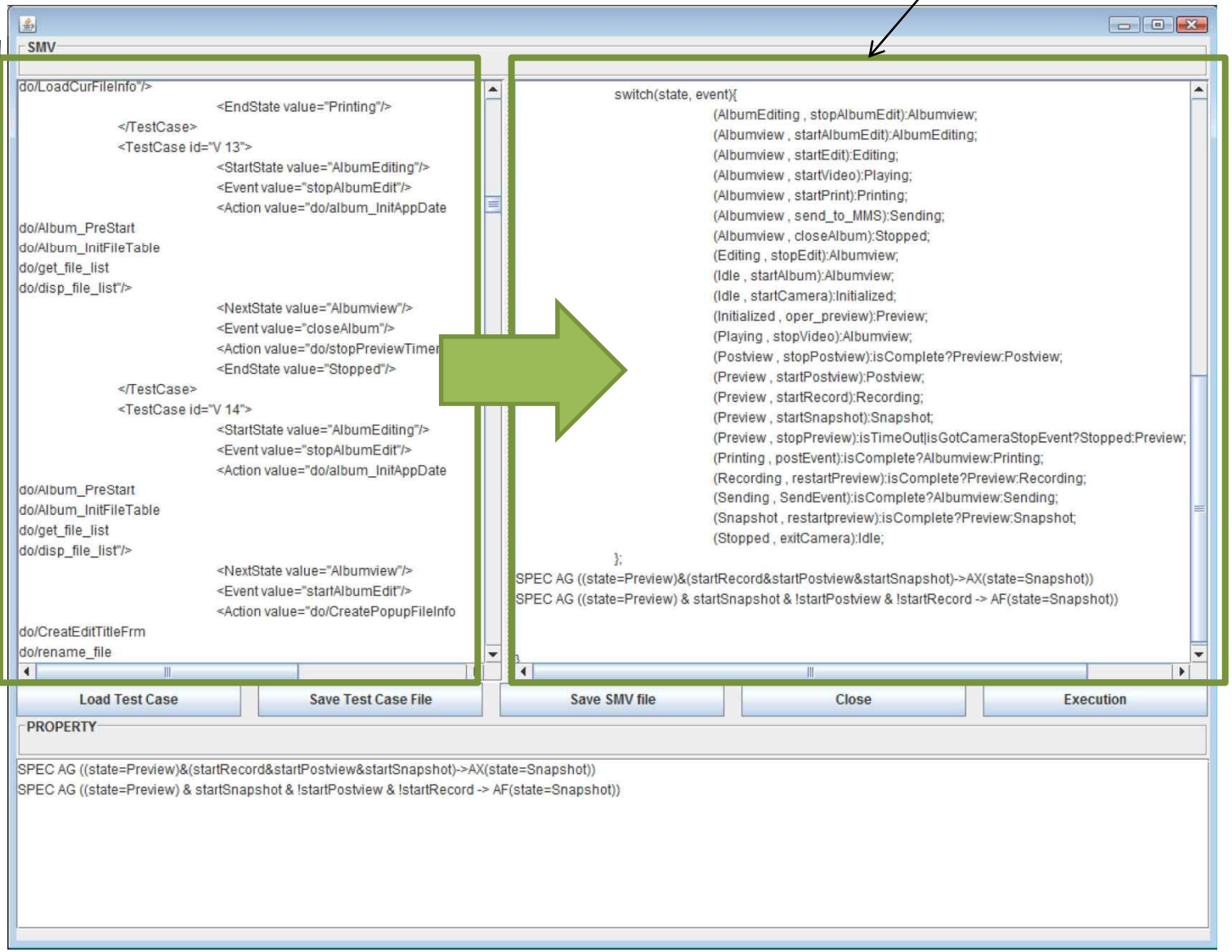
AG(((state=Preview) & startSnapshot & ! startRecord & ! stopPostview) → AF (state=Snapshot))

Test cases (Input)	Expected output
(state = Preview, startSnapshot = 1)	(state = Snapshot)
(state = Preview, startRecord = 1)	(state = Recording)
(state = Preview, startPostview = 1)	(state = Postview)
(state = Preview, stopPreview = 1 , isTimeOut =1) or (state = Preview, stopPreview = 1 , isGotCameraStopEvent = 1)	(state = Stopped)
(state = Snapshot, restartpreview = 1 , isGotCamsensorError = 1) (state = Snapshot, restartpreview = 1 , isSavephoto = OK)	(state = Preview)
(state = Recording, stopRecord = 1 , isDotRecordStopEvent = 1) (state = Recording, stopRecord = 1 , isMemoryFulled = 1) (state = Recording, stopRecord = 1 , isGotCamsensorError = 1)	(state = Preview)
(state = Postview, stopPostview = 1)	(state = Preview)
(state = Stopped, exitCamera = 1)	(state = Idle)

A test suite for the UML specification

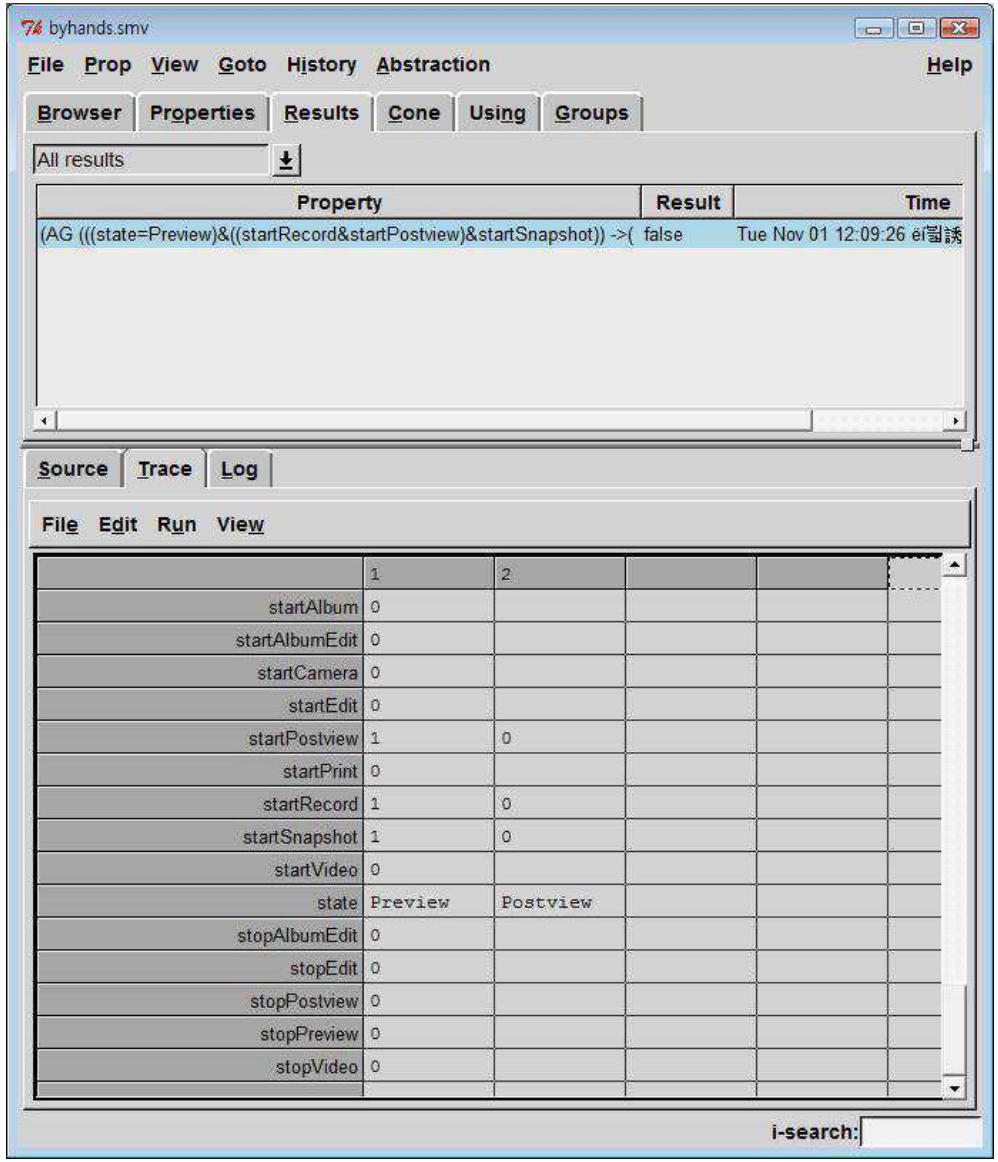
test cases

SMV input program



TC2SMV

(1) → *False*
(2) → *True*



A result of SMV model checking

Safety Requirement (1) :

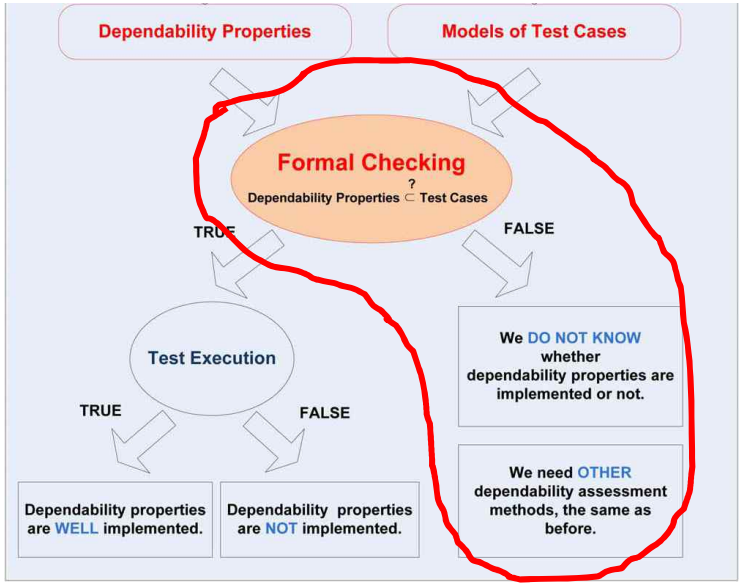
"If the camera button is pressed, then startSnapshot event should be executed at first, even if three events occur simultaneously."

Safety Property (1) : *False*

AG(((state=Preview) & startSnapshot & startRecord & stopPostview) → AX (state=Snapshot))

→ We don't know for now

→ We need other methods to assess it!



Safety Requirement (2) :

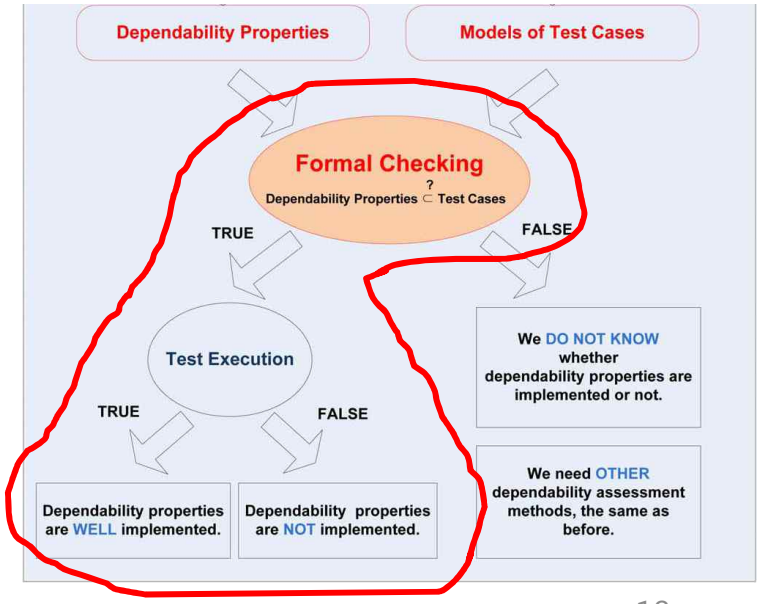
"If the camera button is pressed when the system is ready to take pictures, then it should take a picture, eventually."

Safety Property (2) : *True*

AG(((state=Preview) & startSnapshot & ! startRecord & ! stopPostview) → AF (state=Snapshot))

→ **It may be well implemented (if the test succeeds)**

→ **Safety assessment has been done!**



Needs More Consideration

Formal Checking : Inclusion vs. Satisfaction

- ***Model checking vs. Equivalence checking***
- ***SMV vs. VIS***
- ***SMV input programming language vs. Verilog***

Transformation of safety requirements

- ***Safety analysis → Safety requirements → Safety Properties***

Level of dependability requirements and test cases

- ***Scope of dependability analysis (System vs. Software vs. Component)***
- ***System test vs. Unit test***
- ***Model-based testing vs. Functional testing***

KU KONKUK UNIVERSITY

Thank you

http://dslab.konkuk.ac.kr