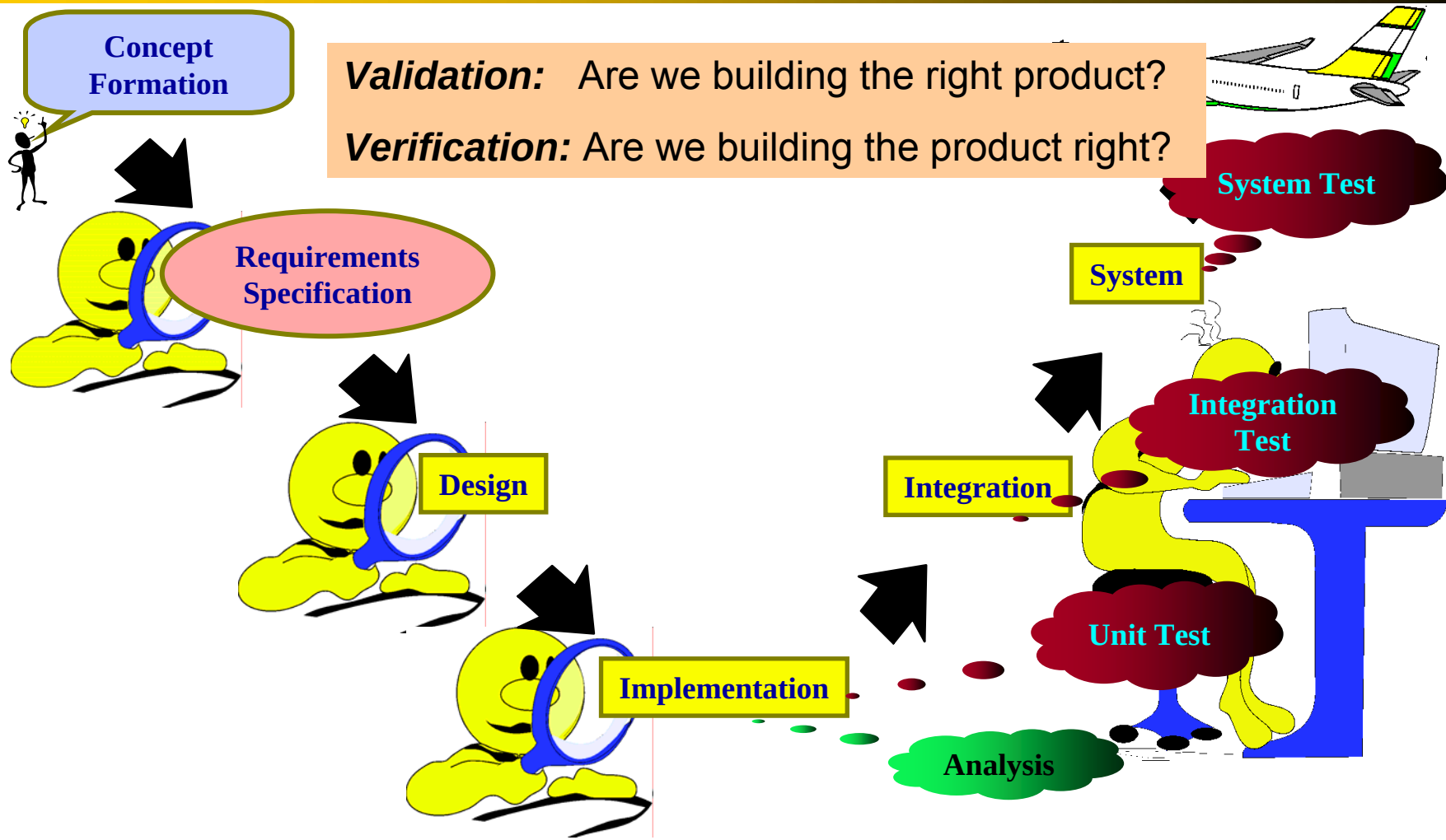
Coverage Metrics for Requirements-Based Testing

Ajitha Rajan

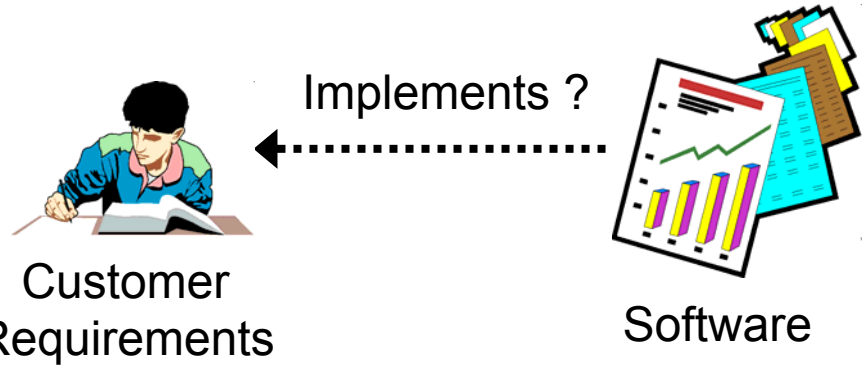
Adviser: Prof. Mats Heimdahl

UMSEC
University of Minnesota
Software Engineering Center

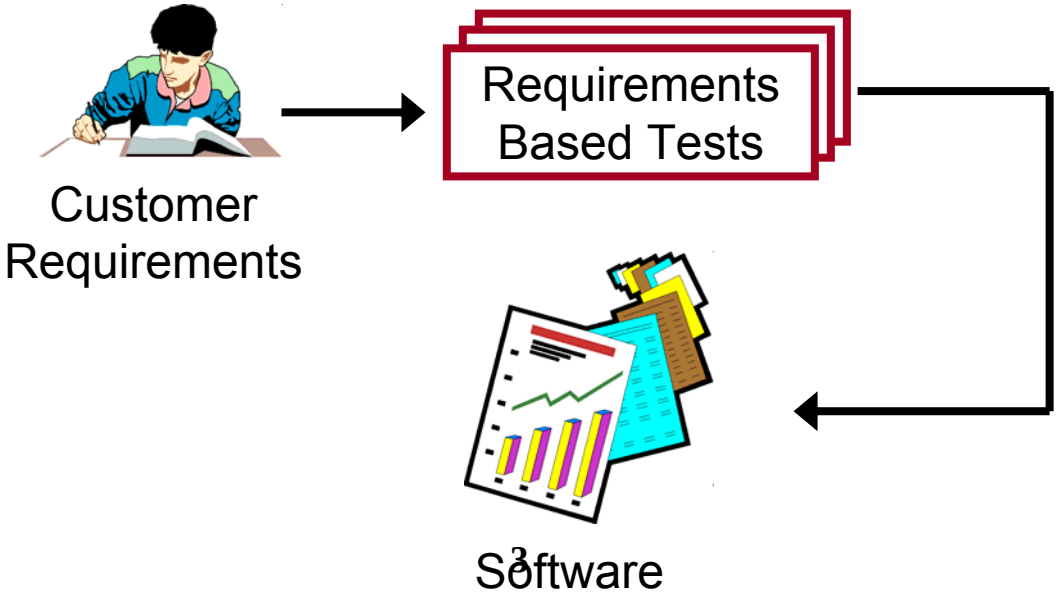
How we develop software



Software Validation



Validation Testing



Black-Box Testing

If Switch is pressed or clap is detected, then Light will turn on

1. Switch = Pressed, Light = On

Input

Expected Output

Does passing this test case indicate that the system has correctly implemented the requirement?

2. Clap_Detected = True, Light = On

3. Switch = Pressed, Clap_Detected = True, Light = On

4.. Switch = Not Pressed, Clap_Detected = False

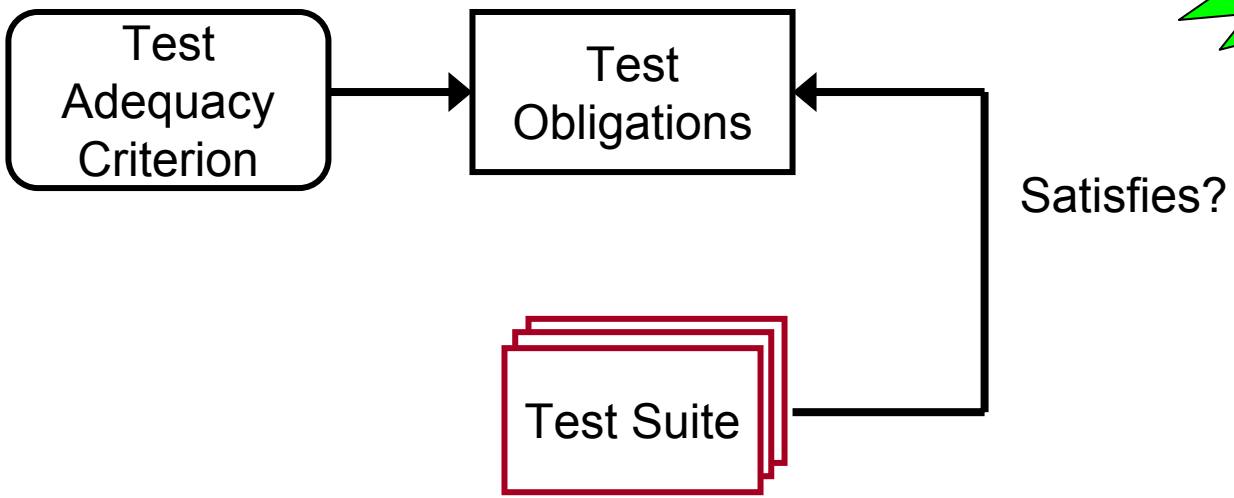
Black-Box Test Adequacy Criteria

Ideally : *Exhaustively* test all possible behaviors specified in the requirement.

Infeasible

Practically: *Adequately* test behaviors specified in the requirement.

Test Adequacy Criterion

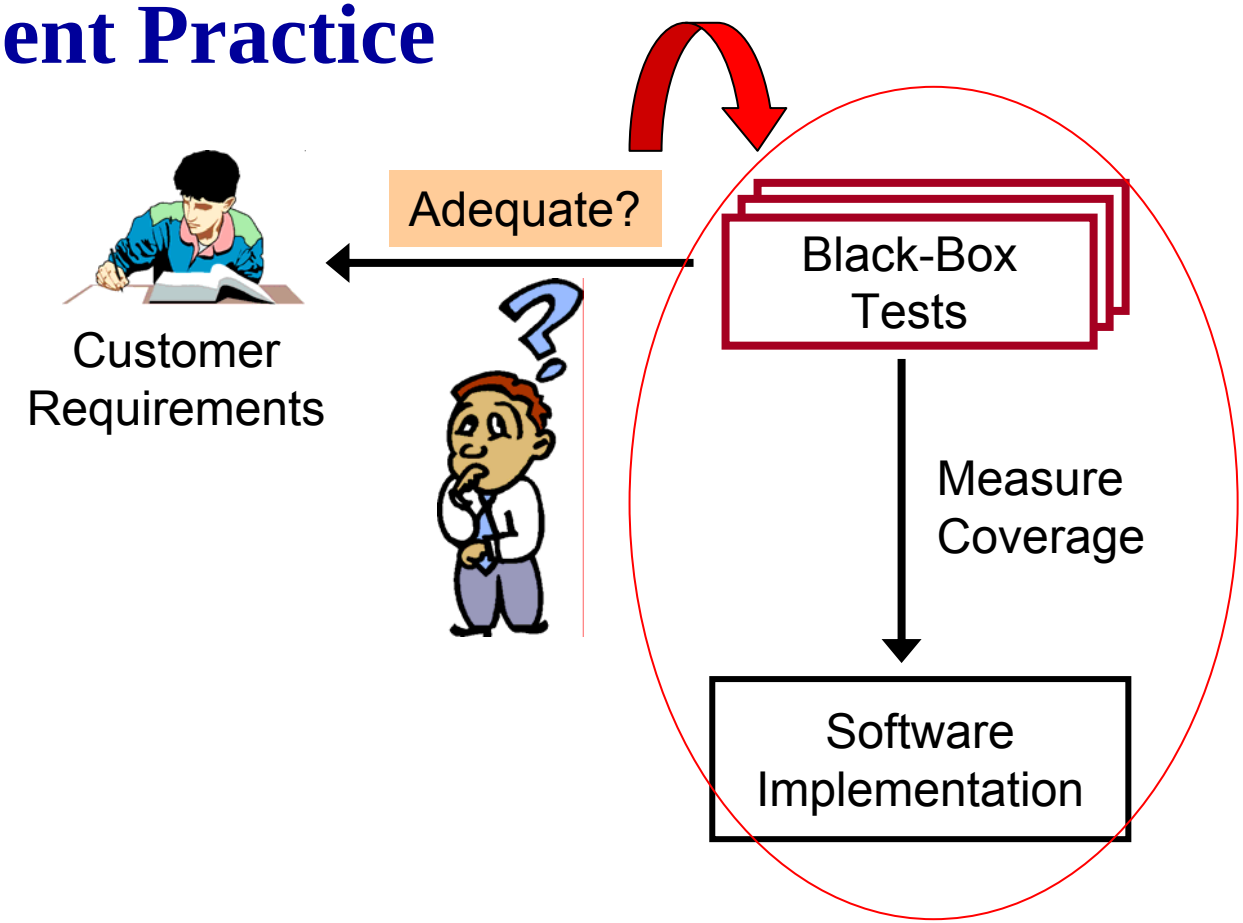


Dissertation Question

How can we *objectively* assess whether or not the black-box tests adequately cover the behaviors specified in the requirements?

Adequacy of Black-Box Test Suites

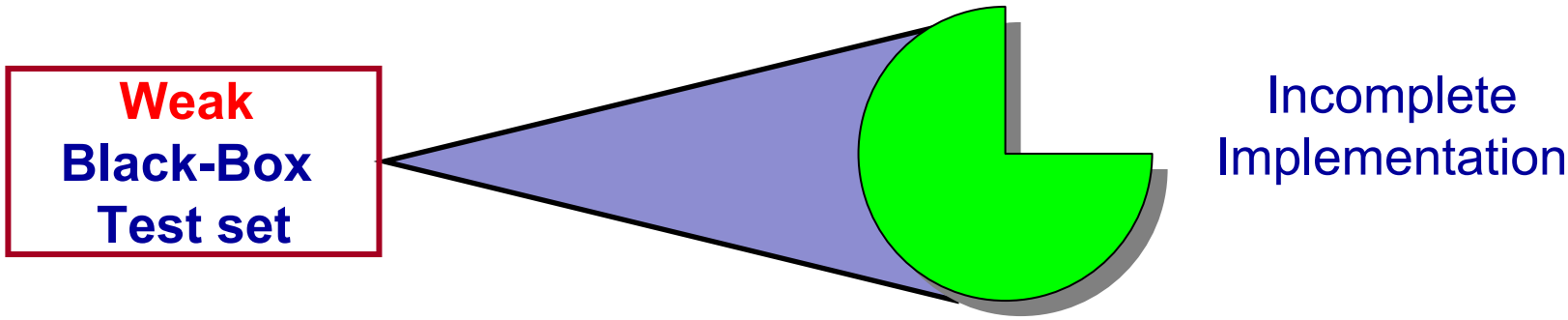
Current Practice



Adequacy of Black Box Test Suites

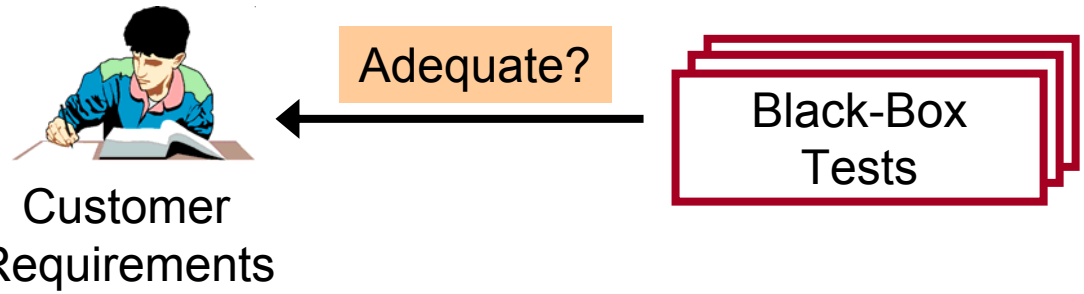
Problems with current practice

- Indirect measure
  - Defects of omission in implementation not exposed.



- Implementation is necessary
  - adequacy can only be determined late in the development process

Dissertation Contribution



Requirements Coverage Metrics: Provide objective, implementation-independent measures of how well a black-box test suite exercises requirements

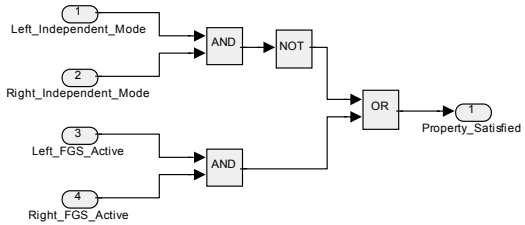
The Idea

Write requirements in a formal notation…

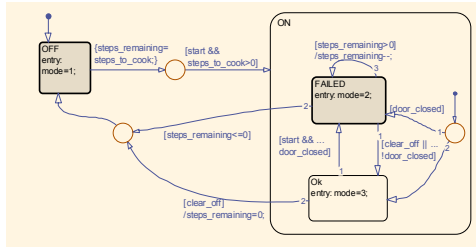
G (FD_On -> Cues_On);

G((¬ Onside_FD_On ∧ ¬ Is_AP_Engaged) → X(Is_AP_Engaged → Onside_FD_On))

Temporal Logic



Synchronous Observers

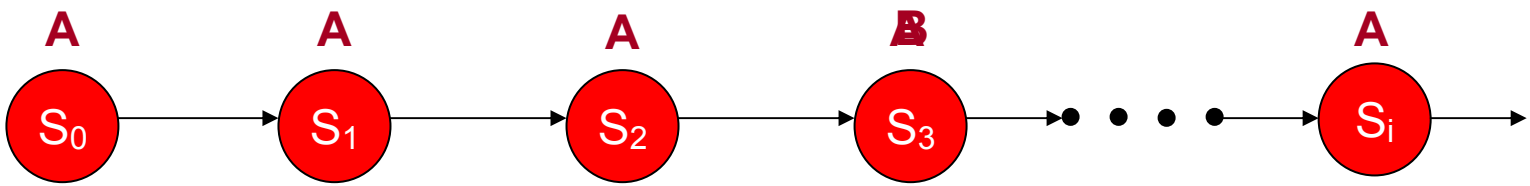


State Machines

…then define structural coverage metrics to directly and objectively describe coverage of requirements

LTL Temporal Operators

Operator	Notation	Meaning
Globally A	G(A)	Formula *A* is true in all states
Future A	F(A)	Formula *A* is true in some future state
A until B	A U B	Formula *A* is true in every state until *B* becomes true. *B* must eventually become true for the property to be true.
Next A	X(A)	Formula *A* is true in the next state



Formalizing Requirements

"If the onside FD cues are off, the onside FD cues shall be displayed when the AP is engaged"

$$G((\neg\ Onside\_FD\_On\ \wedge\ \neg\ Is\_AP\_Engaged) \rightarrow X(Is\_AP\_Engaged \rightarrow Onside\_FD\_On))$$

- Possible Coverage Metrics
  - Requirements coverage: Single test case that demonstrates that req. is satisfied
    - Prone to "dumb" tests, e.g. execution in which AP is never engaged.
  - More rigorous metric is necessary

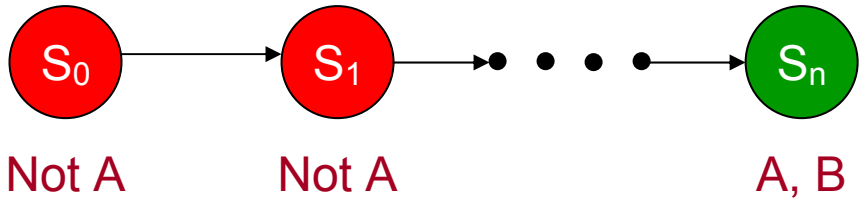
Task - 1

- Define a collection of requirements coverage criteria
- Formalize the requirements coverage obligations

Requirements Antecedent Coverage

- Many of the requirements in the FGS are of the form :
  - *Globally if 'A' occurs then 'B' will occur*

    G (A → B)
  - Two ways of satisfying (A → B)
    - A is false
    - A is true and B is true
- Requirements Antecedent Coverage – Test cases will exercise the antecedent.



Black-Box Testing

If Switch is pressed or clap is detected, then Light will turn on

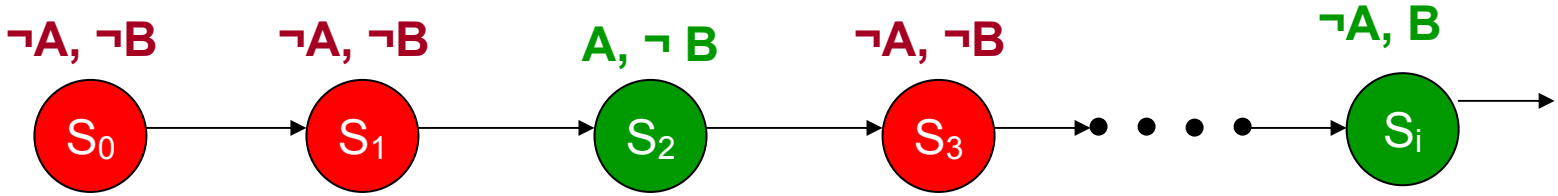
G ((Switch = Pressed) V (Clap_Detected = True) → (Light = On))

1. Switch = Pressed, Light = On
2. Clap_Detected = True, Light = On
3. Switch = Pressed, Clap_Detected = True, Light = On

Unique First Cause (UFC) Coverage

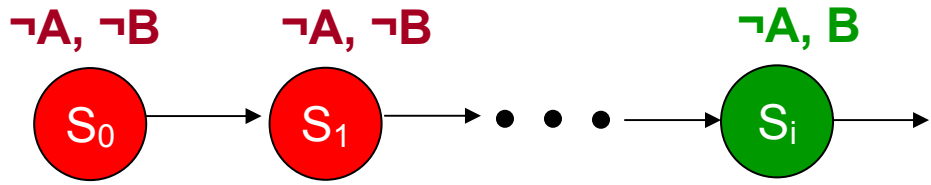
"System shall eventually generate an *Ack* or a *Time Out*"

Req. LTL property - $F(A \vee B)$.



Path satisfies UFC obligation for A but not B.

To show independence of B,



Formal UFC obligation for A : $\neg(A \vee B)\ U\ (A \wedge \neg B)$

for B : $\neg(A \vee B)\ U\ (B \wedge \neg A)$

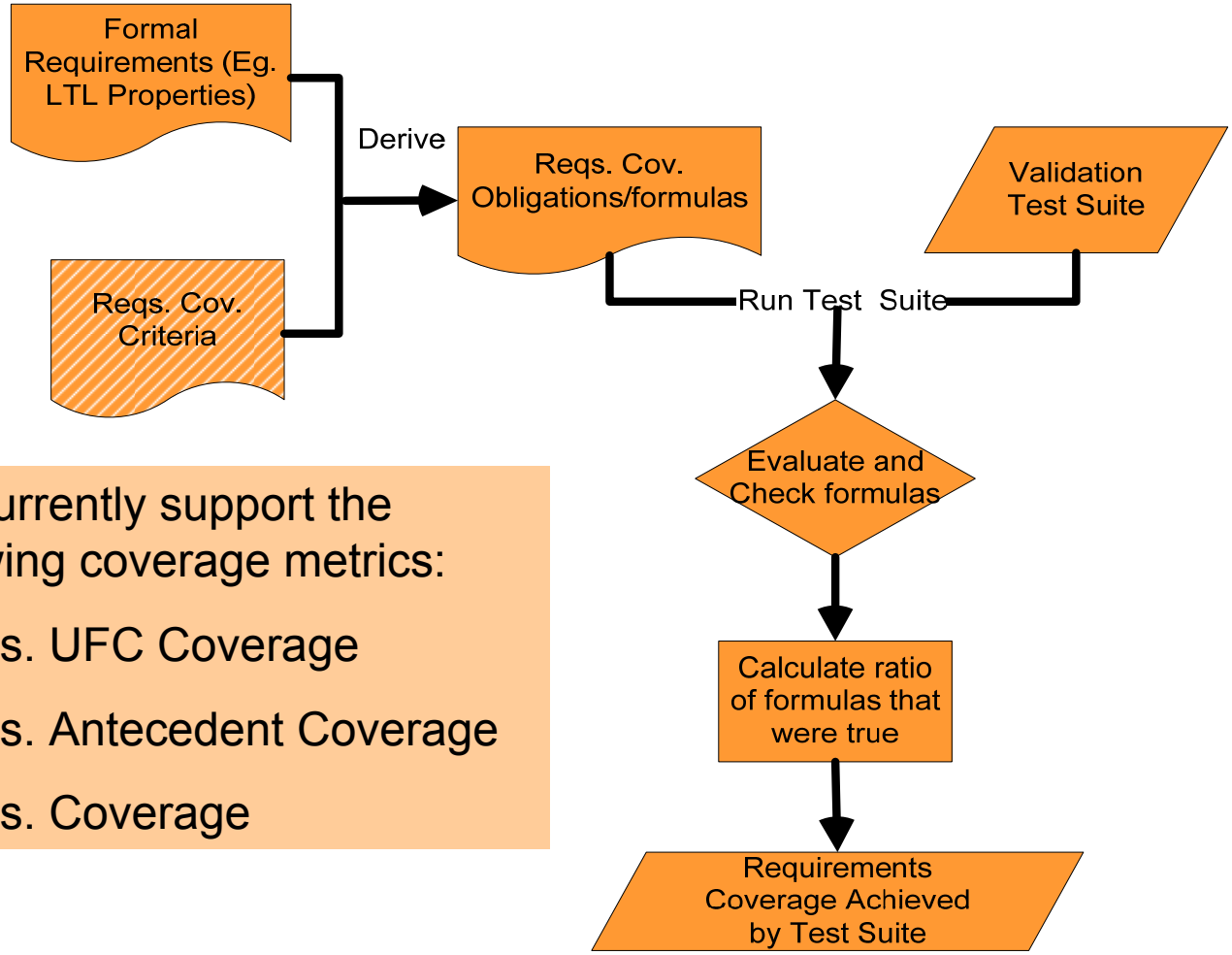
UFC Coverage

- $G(A)^+ = \{A \, U \, (a \wedge G(A)) \mid a \in A^+\}$
  $G(A)^- = \{A \, U \, a \mid a \in A^-\}$
- $F(A)^+ = \{\neg A \, U \, a \mid a \in A^+\}$
  $F(A)^- = \{\neg A \, U \, (a \wedge G(\neg A)) \mid a \in A^-\}$
- $(A \, U \, B)^+ =$
  $\{(A \wedge \neg B) \, U \, ((a \wedge \neg B) \wedge (A \, U \, B)) \mid a \in A^+\} \cup$
  $\{(A \wedge \neg B) \, U \, b \mid b \in B^+\}$

  $(A \, U \, B)^- =$
  $\{(A \wedge \neg B) \, U \, (a \wedge \neg B) \mid a \in A^-\} \cup$
  $\{(A \wedge \neg B) \, U \, (b \wedge \neg(A \, U \, B)) \mid b \in B^-\}$
- $X(A)^+ = \{X(a) \mid a \in A^+\}$
  $X(A)^- = \{X(a) \mid a \in A^-\}$

Michael Whalen, Ajitha Rajan, Mats Heimdahl and Steven Miller. **Coverage Metrics for Requirements-Based Testing.** In *Proceedings of ISSTA 2006.*

Validation Test Adequacy Measurement Tool



We currently support the following coverage metrics:

- Reqs. UFC Coverage
- Reqs. Antecedent Coverage
- Reqs. Coverage

Case Study

Interested in determining:

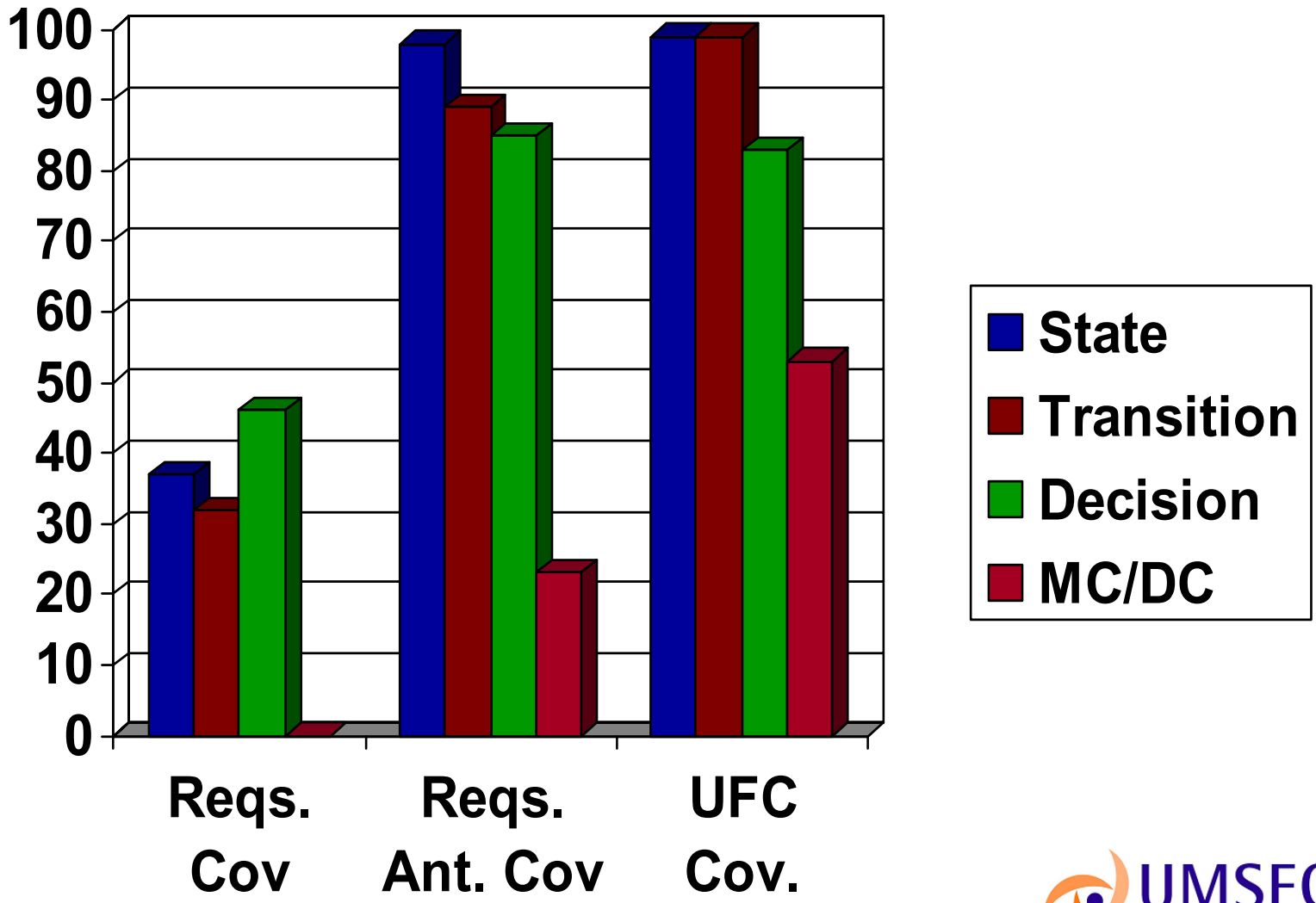
- Feasibility of generating such tests
- Number of test-cases needed to provide requirements coverage for a realistic example
- What coverage of the model would these test-sets provide?

Case Study

The requirements-based test-suites for the FGS case example were generated in 3 ways:

1. Requirements Coverage
2. Requirements Antecedent Coverage
3. UFC coverage

Results and Analysis



Results and Analysis

- Test-suite generated for UFC gave very low MC/DC over the model

  "When the FGS is in independent mode, it shall be active".

  G(m_Independent_Mode_Condition.result → X(Is_This_Side_Active = 1))

RSML^{-e} Macro

Independent_Mode_Condition = ((Is_LAPPR_Active & Is_VAPPR_Active & Is_Offside_LAPPR_Active & Is_Offside_VAPPR_Active) | (Is_VGA_Active & Is_Offside_VGA_Active))

Structure of *Independent_Mode_Condition* is not captured in the property

Summary

Several benefits of defining requirements adequacy criteria:

- *Direct measure* of how well a black-box test suite addresses a set of requirements
- *Implementation independent* assessment of the adequacy of a black-box test suite
- Means for measuring the *adequacy of requirements* on a given implementation

Questions