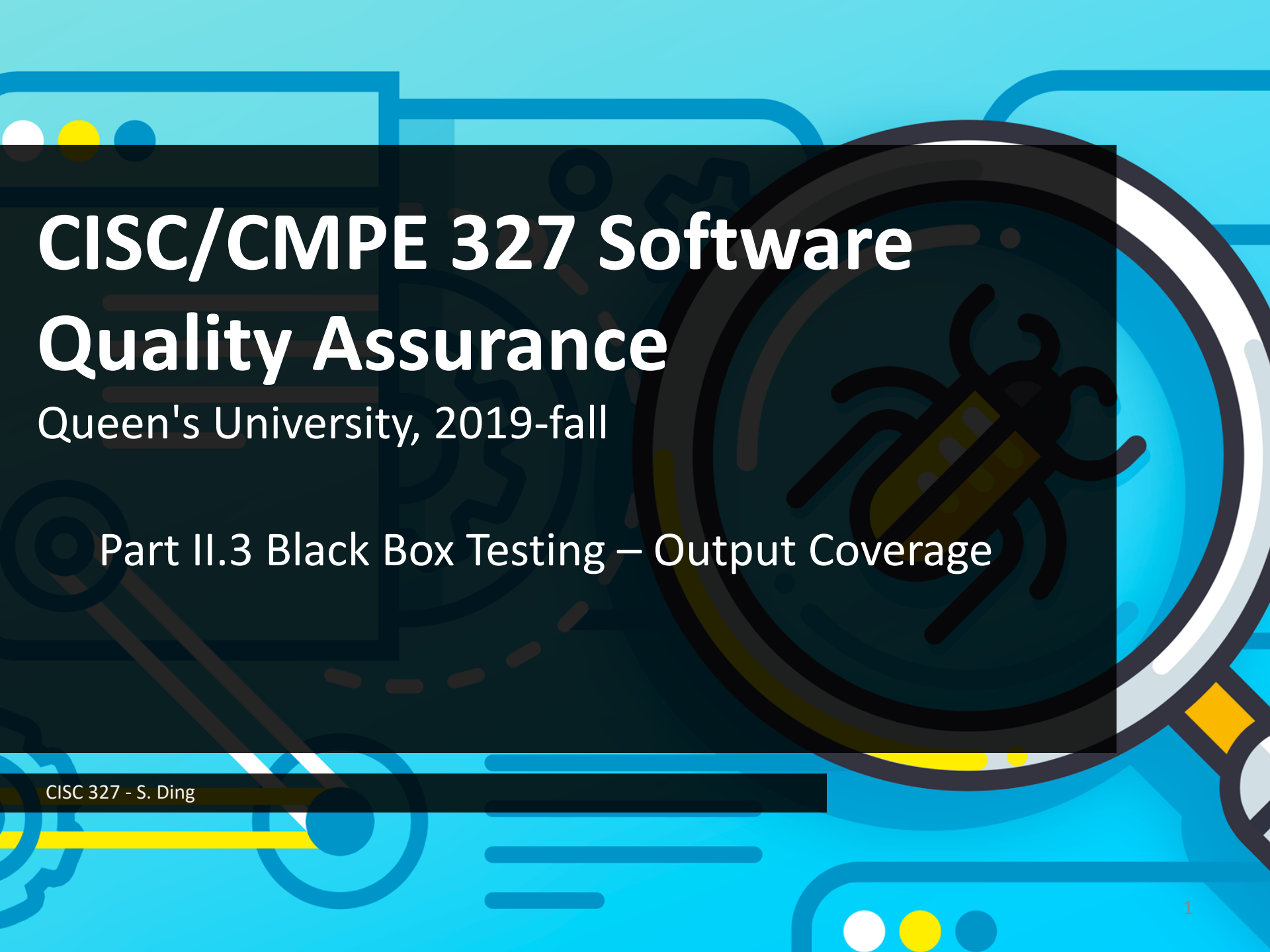


CISC/CMPE 327 Software Quality Assurance

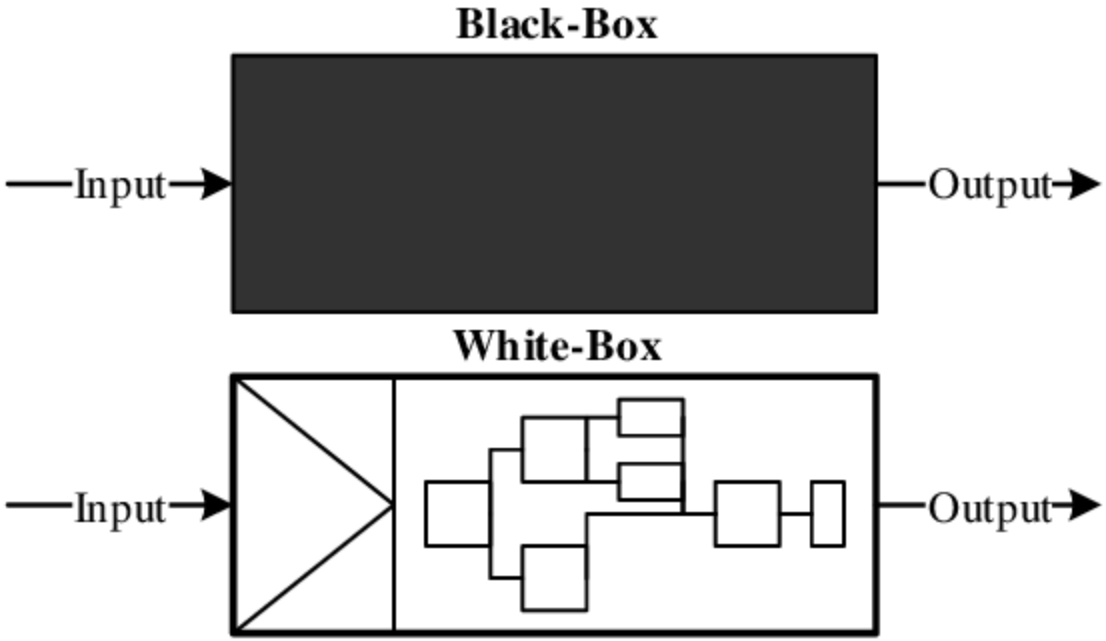
Queen's University, 2019-fall

Part II.3 Black Box Testing – Output Coverage

Black Box Testing

- Outline
  - input coverage testing
  - output coverage testing
  - Output coverage methods
    - exhaustive
    - output partitioning
  - Testing multiple input or output streams
  - Model-based testing

Black Box Testing



Output?

- all the possible outputs specified in the functional specification (requirements)

Output Coverage Testing

1. Analyze all the possible outputs
2. Create tests to cause each one

output -> input

"Given as input two integers x and y, output all the positive numbers smaller than or equal to x that are evenly divisible by y. If either x or y is zero, then output zero."

Output: 3, 6, 9, 12

x: ???

y: ???

Output Coverage Testing

- More difficult than input coverage
- Effective:
  - > finding problems
  - > develop deep understanding of the requirements

Exhaustive Output Testing

- Test them all !
- requirements say:
  - "Output 1 if two input integers are equal, 0 otherwise"

Exhaustive Output Testing

- Test them all !
- requirements say:
  - "Output 1 if two input integers are equal, 0 otherwise"

- Only two test cases:
  - Output 1, output 0

V. S. Input Coverage - Exhaustive

- Test them all !
- requirements say:
  - "Output 1 if two input integers are equal, 0 otherwise"

V. S. Input Coverage - Partition

- Test them all !
- requirements say:
  - "Output 1 if two input integers are equal, 0 otherwise"

V. S. Input Coverage - Partition

- Test them all !
- requirements say:
  - "Output 1 if two input integers are equal, 0 otherwise"
  - Numbers equal, numbers not equal, first number zero / positive / negative, second number zero / positive / negative

Exhaustive Output Testing

- More practical than Exhaustive Input Testing?
  - Exhaustive output testing makes one test for every possible output
  - Practical more often than input testing
  - But still impractical in general
    - an infinite number of different possible outputs

Output Partitioning

- Partition all the possible outputs into a set of equivalence classes with something in common

Output Partition Testing

"Given as input two integers x and y, output all the positive numbers smaller than or equal to x that are evenly divisible by y. If either x or y is zero, then output zero."

– The output is a list of integers, so we might partition into the following cases:

Number of integers in output

output values	zero	one	many
all positive	P1	P2	P3

Output Partition Testing: Designing Inputs

- Design inputs to cause outputs in each partition
- This is difficult and time-consuming
  - The biggest drawback to output coverage testing!

- We cannot find such an input
  - This implies an error or oversight in either the requirements or in the partition analysis

Multiple Input or Output Streams

- A Separation of Concerns
  - Multiple inputs (variable, file, socket etc.)
  - Must create separate coverage tests for each one
  - Effectively, what we do is treat each separate file or stream as a pre-made input or output partition, within which we make a separate set of smaller partitions

Black Box Testing at Different Levels

- levels of testing
  - Unit/Integration/System

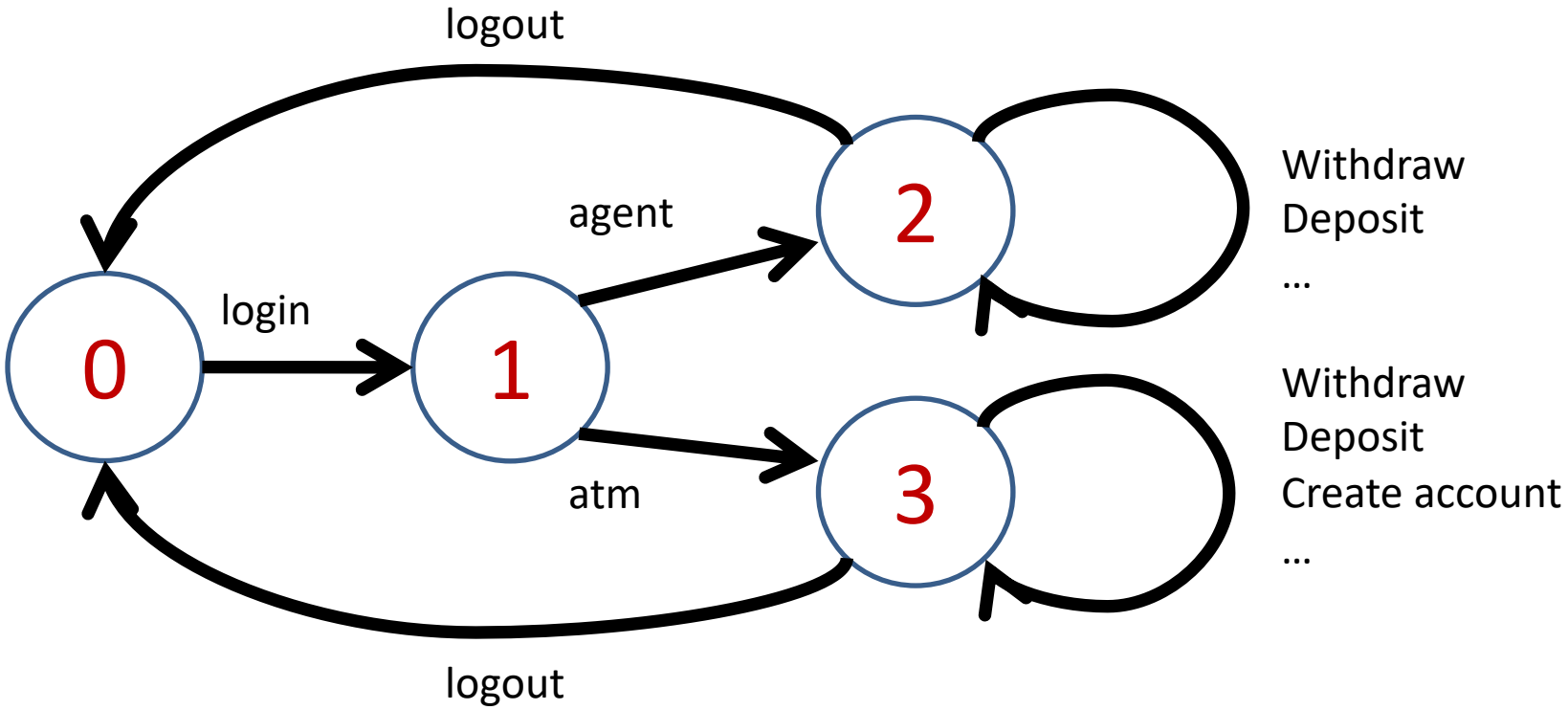
- In particular, black box testing of all kinds can be used at every level of software development

Model-Based Testing

- Model-Driven Engineering (MDE)
  - A modern new black-box method is model-based testing, part of MDE
  - Model-based testing does not use a specification, but rather a formal state model of the process implemented by the program
  - State models are high-level abstractions (simplifications) of the program's intent, usually expressed at the level of the problem domain rather than the computer
  - State models ignore implementation details, but retain essential states of the process

Model-Based Testing

- Model-Driven Engineering
  - For example, the following might be a state model of the login aspect of the Front End



Model-Driven Engineering

- Models are formal (mathematical) specifications of the process to be implemented
- Formal models can be used in several ways
  - To verify that the model (formal specification) is itself correct, using model checking (NASA, Airbus) (CISC 422)
  - To generate some or all of the implementation from the formal model, if it is detailed enough (General Motors)
  - To test that the implementation is consistent with the formal model (model-based testing)

Model-Based Testing

- Advantages:
  - Automatic test generation
  - Tests against a formal specification (the verified model)
  - Covers all essential behaviour
  - Still a black box method, with all its advantages
    - Requires only the model, not the code
  - Yields high confidence in the correctness of the final code

Model-Based Testing

- Disadvantages:
  - Heavyweight test method, probably only practical for safety-critical and security-critical applications (aerospace, automotive, etc.)

Summary…

- Black Box Testing
  - Output coverage methods analyze the set of possible outputs specified and create tests to cover them
  - Exhaustive output testing and output partitioning are similar but distinct from input coverage methods
  - Multiple input or output streams / files are handled by treating them as a predefined partitioning boundary

...Summary.

- Black Box Testing
  - We can also apply black box methods at lower levels of testing, if we have the architecture or detailed design
  - Model-driven engineering (MDE) can assist to automatically generate high quality tests using model-based testing