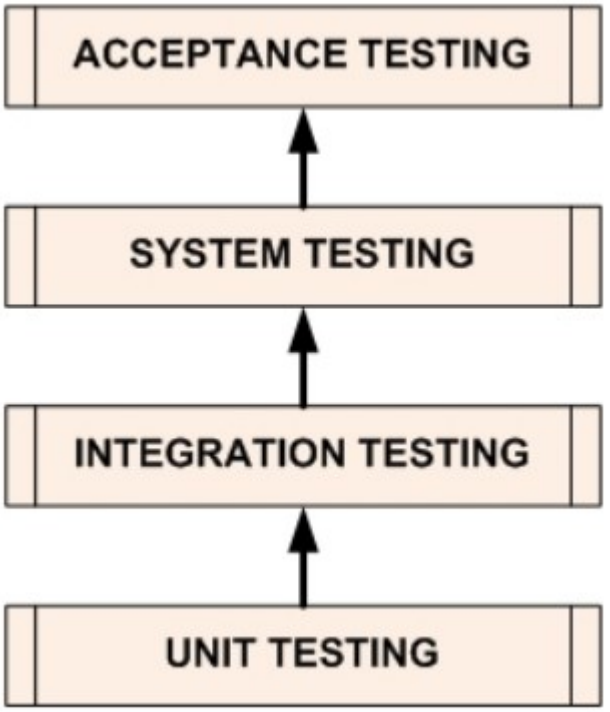
Software Testing Levels

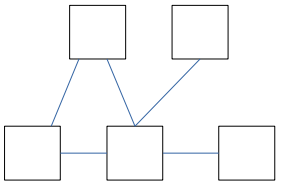
- There are generally four levels of tests:

  Unit >> Integration >> System >> Acceptance Testing.

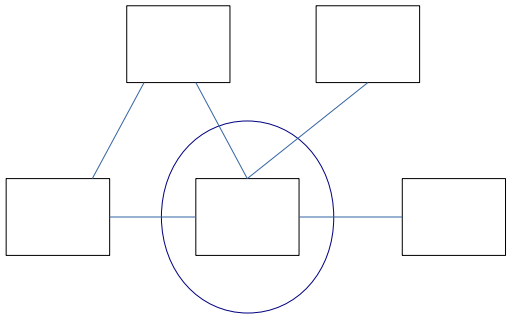


Software Testing Levels

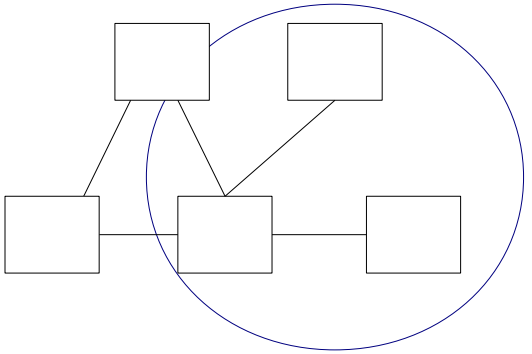
- Lets consider a software system.
  - A system made out of components that interact with one another



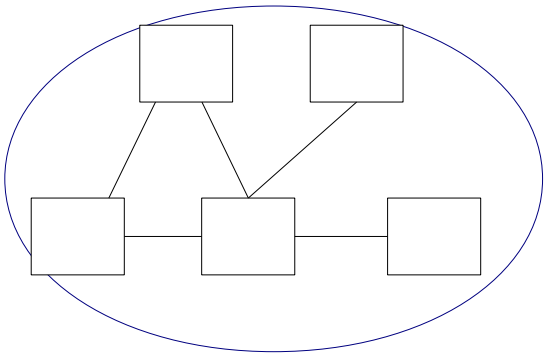
- The first level we consider in testing is called **unit testing**, which is the testing of an individual unit or module in isolation. The purpose is to validate that each unit of the software performs as designed.
- The next level is to consider a multiple modules and their interactions, which is called **integration testing**. The purpose of this level of testing is to expose faults in the interaction between integrated units.
- The next step is to test the complete system as whole, which is called **system testing**. The purpose of this test is to evaluate the system's compliance with the specified requirements.



Unit Testing



Integration Testing



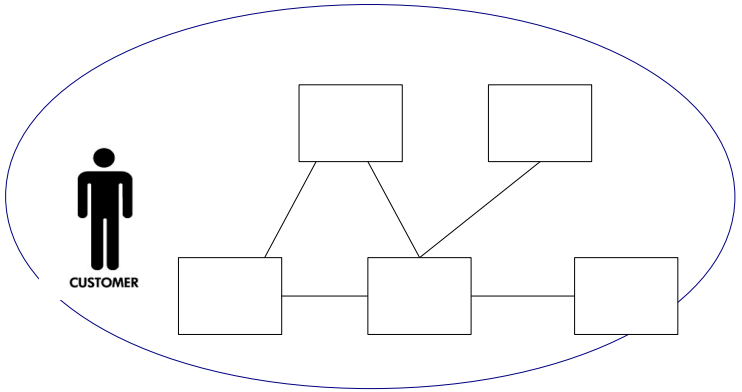
System Testing

Oregon State University

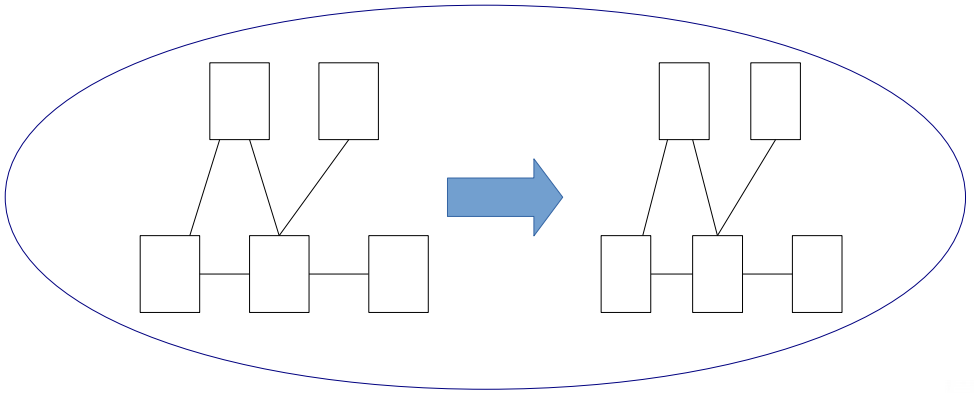
Software Testing Levels

- **Acceptance Testing** which is the validation of the software against the customer requirements. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery.

- **Regression Testing** is a type of testing
  - We perform regression testing every time that we change our system
  - We need to make sure that the changes behave as it is intended and the unchanged code is not negatively affected by these changes/modifications



Acceptance Testing



Regression Testing

Oregon State University

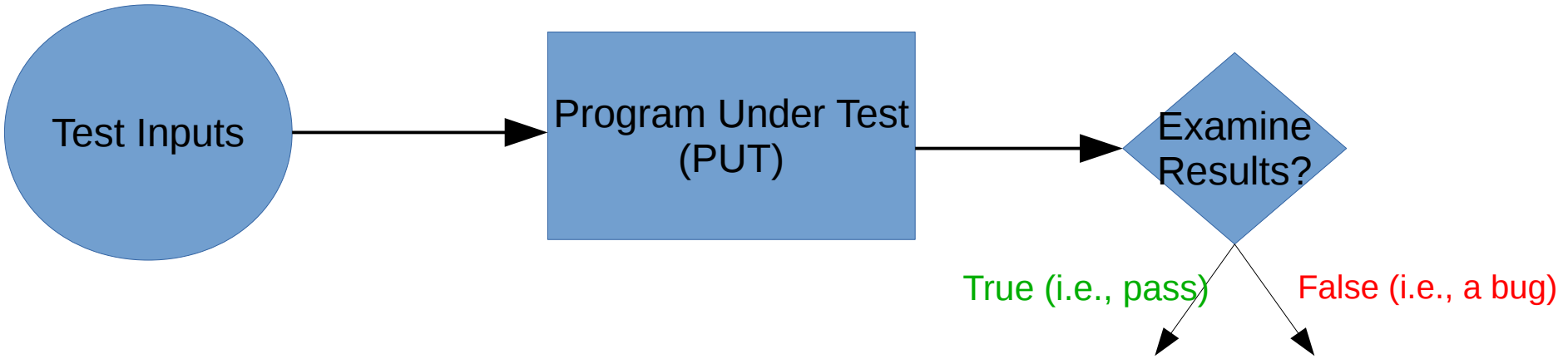
UNIT TESTING

- **Unit testing:** The testing of individual software components. Unit testing is essentially done by the programmer.
- **Benefits** of unit testing:
  - We can test parts of a project without waiting for the other parts to be completed.
  - If a test fails, it directly points out the cause.
  - Debugging is simplified and less complicated.
  - Cost of fixing a defect identified during the early stages is less compared to that during later stage.
- Unit Testing **Techniques**
  - Functional Testing Techniques (**Black-Box**)
    - Choose test inputs ***without*** looking at implementation
  - Structural Techniques (**White-Box**)
    - Choose test inputs ***with*** knowledge of implementation

Oregon State University

How is testing done?

1) **Choose input data**
2) Define the expected outcome
3) Run program under test against the input and record the results
4) **Examine results against the expected outcome**

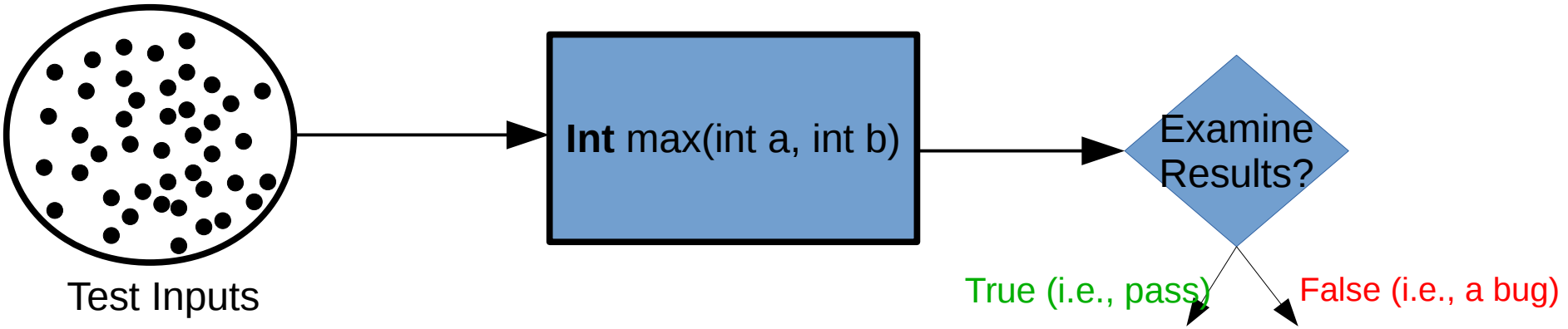


Oregon State University

What is So Hard about Testing?

Example: Lets consider a program under test (**PUT**) that takes two integer and returns the maximum value.

```
int max(int a, int b)
// effects: a > b => returns a
// a < b => returns b
// a = b => returns a
```



The question is: How can we select a set of inputs from input domain (i.e., test inputs) for our **PUT** that after we run them, we have enough confidence that the PUT is implemented correctly?

References

Pezze + Young, “Software Testing and Analysis”, Chapter 10 & 11
Patton, Ron. "Software Testing." (2000). Chapter 4 & 5
Sommerville, I., Software Engineering, Sixth Edition, Addison-Wesley, 2001 Chapter 20

Oregon State
University