University of Sheffield

COM3529 Software Testing and Analysis

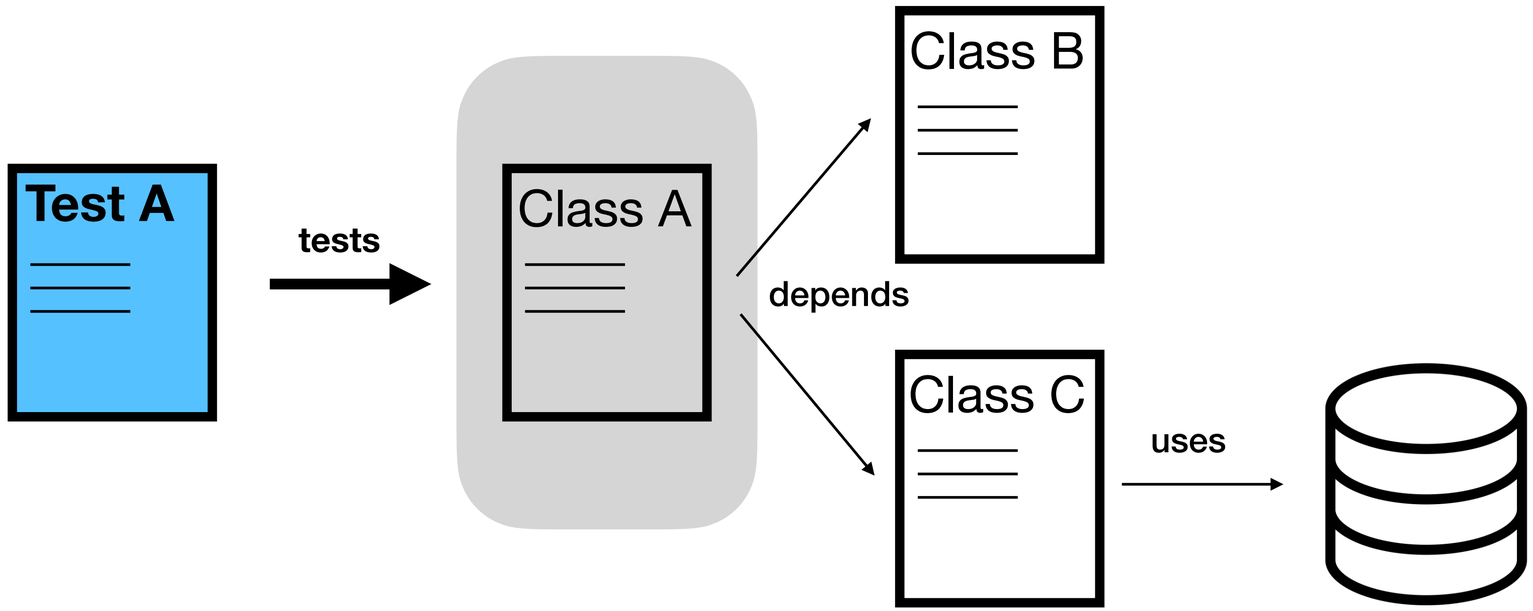
Test Scope

Professor Phil McMinn

Unit Testing

A unit is an individual component of a system, such as a class or an individual method.

Testing units in isolation is called **unit testing**.



Unit Testing

A unit is an individual component of a system, such as a class or an individual method.

Testing units in isolation is called **unit testing.**

- **Fast**
- **Easy to control**
- **Easy to write**

- **Lack reality**
- **Cannot catch all bugs**
  (e.g. interactions with other components or services)

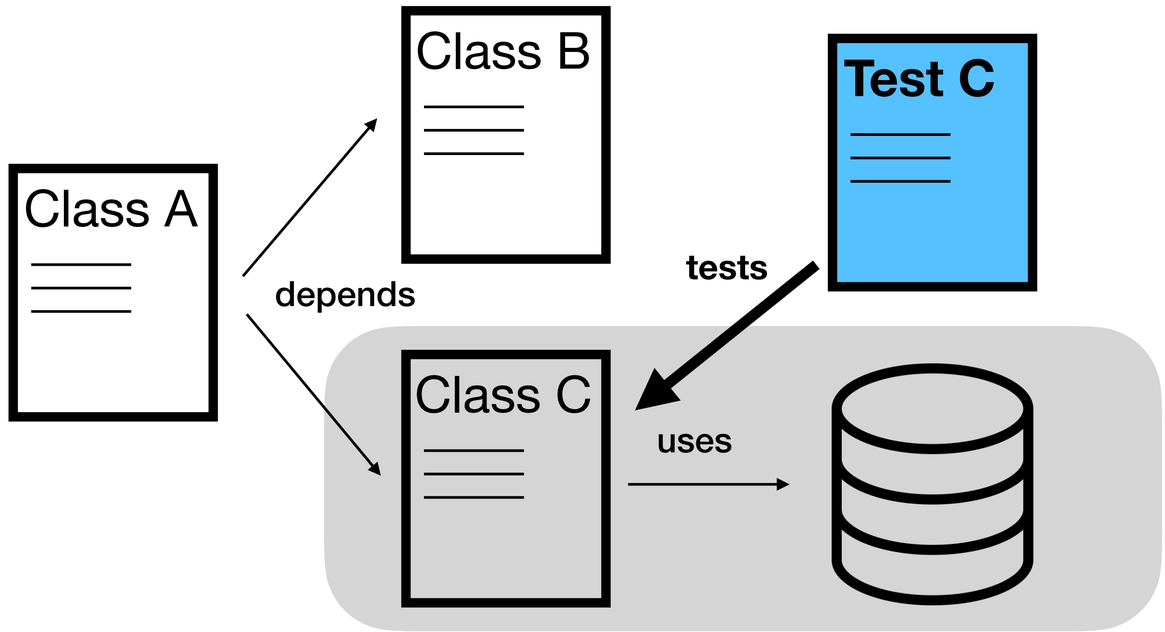
Unit tests are a very useful type of test but are often insufficient on their own.

Integration Tests

Testing in isolation is not enough. Sometimes code goes “beyond” the system’s borders and uses other (often external) components – for example, a database.

Integration tests test the integration between our code and that of external parties.

Example: Testing methods that access a database via SQL queries. Do our methods obtain the right data from the database?



Integration Tests

Testing in isolation is not enough. Sometimes code goes “beyond” the system’s borders and uses other (often external) components – for example, a database.

Integration tests test the integration between our code and that of external parties.

Example: Testing methods that access a database via SQL queries. Do our methods obtain the right data from the database?

- **Can capture integration bugs**
- **Less complex than writing a system test that goes through the entire system, including components we do not care about**

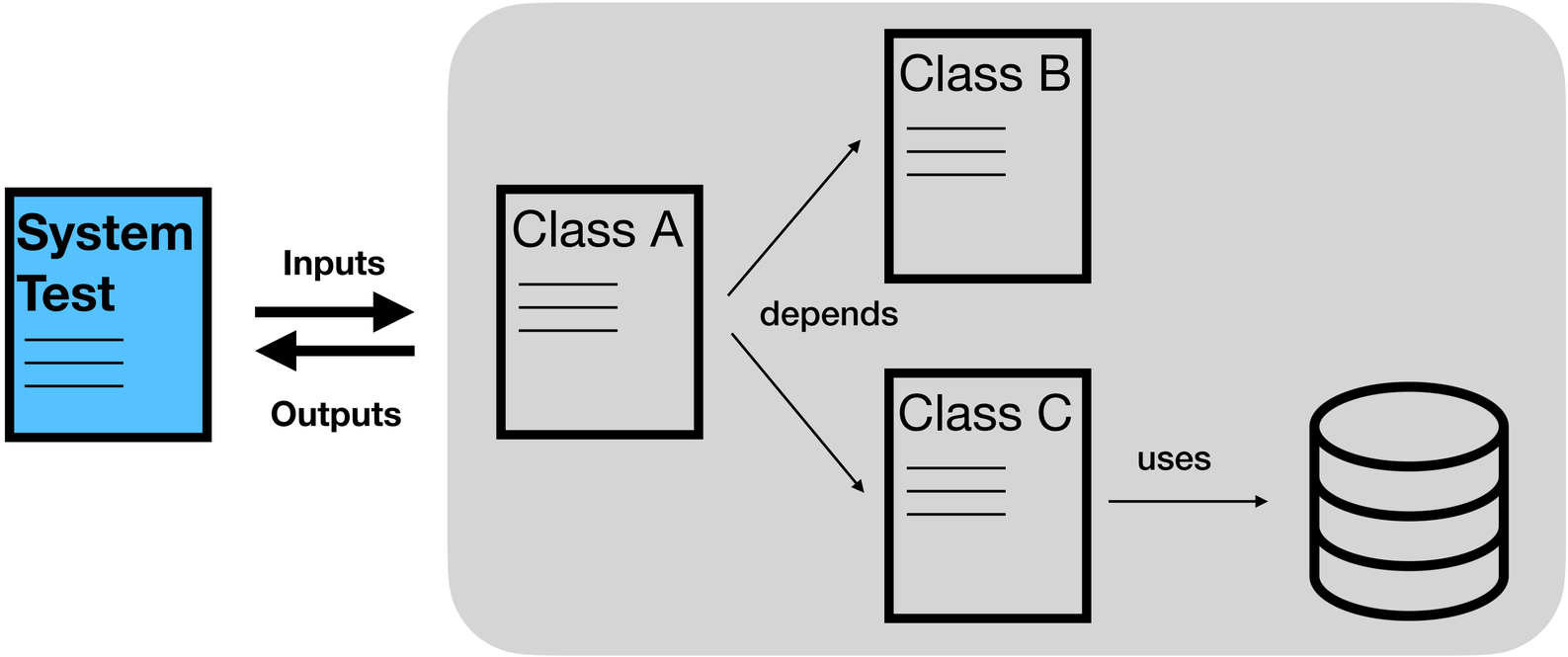
- **Hard to write**, for example:
  - Need to use an isolated instance of the database
  - Put it into a state expected by the test
  - Reset the state afterwards

System Tests

To get a more realistic view of the software we should also perform more realistic tests with it – with all its database, front-end, and other components.

We do not care about how the system works from the inside.

We care that given certain inputs, certain outputs are provided by the system.



System Tests

To get a more realistic view of the software we should also perform more realistic tests with it – with all its database, front-end, and other components.

We do not care about how the system works from the inside.

We care that given certain inputs, certain outputs are provided by the system.

- **Realistic**
  (when the tests perform similarly to the end user, the more confident we can be that the system will work correctly for all end users)

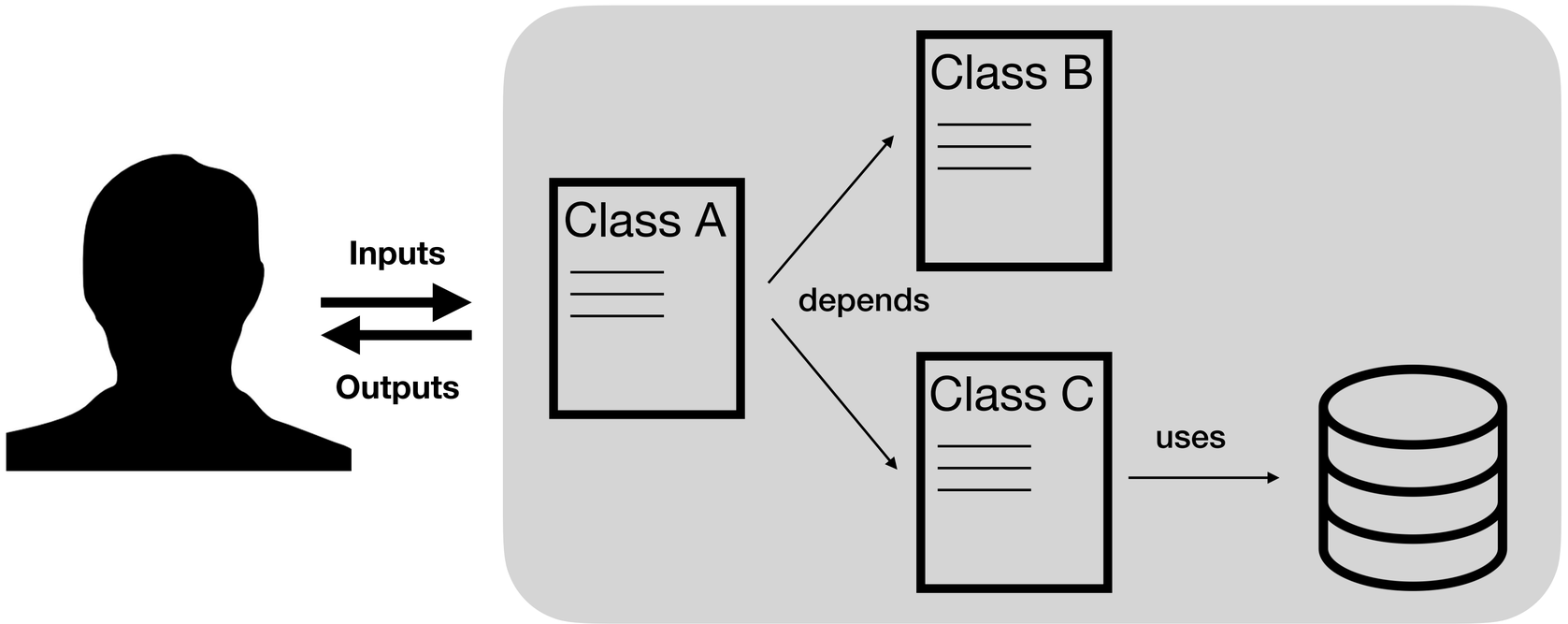
- **Slow!**
- **Hard to write**
  (lots of external services to account for)
- **Prone to Flakiness**

Manual Tests

Not everything can be tested easily in an automated fashion, particularly where there are qualitative judgements (e.g., the quality of a search engine's results).

Furthermore, we may need to explore real system behaviour to know what automated tests to write.

Manual tests are system tests performed manually by a human.



Manual Tests

Not everything can be tested easily in an automated fashion, particularly where there are qualitative judgements (e.g., the quality of a search engine's results).

Furthermore, we may need to explore real system behaviour to know what automated tests to write.

Manual tests are system tests performed manually by a human.

- **Real**
  (The tester is acting as an end-user, actually using the system)

- **Time-consuming**
- **Difficult to reproduce**
- **Tedious**

The Test Triangle