Automock: Interaction-Based Mock Code Generation

Sabrina de F. Souto, UFCG, Brazil
sabrina@dsc.ufcg.edu.br

Supervisor: Dalton Serey, UFCG, Brazil
dalton@dsc.ufcg.edu.br

LADC
Fourth Latin-American Symposium
on Dependable Computing

Student Forum

September 1 – 4
João Pessoa, Paraíba, BRAZIL

2009

Outline

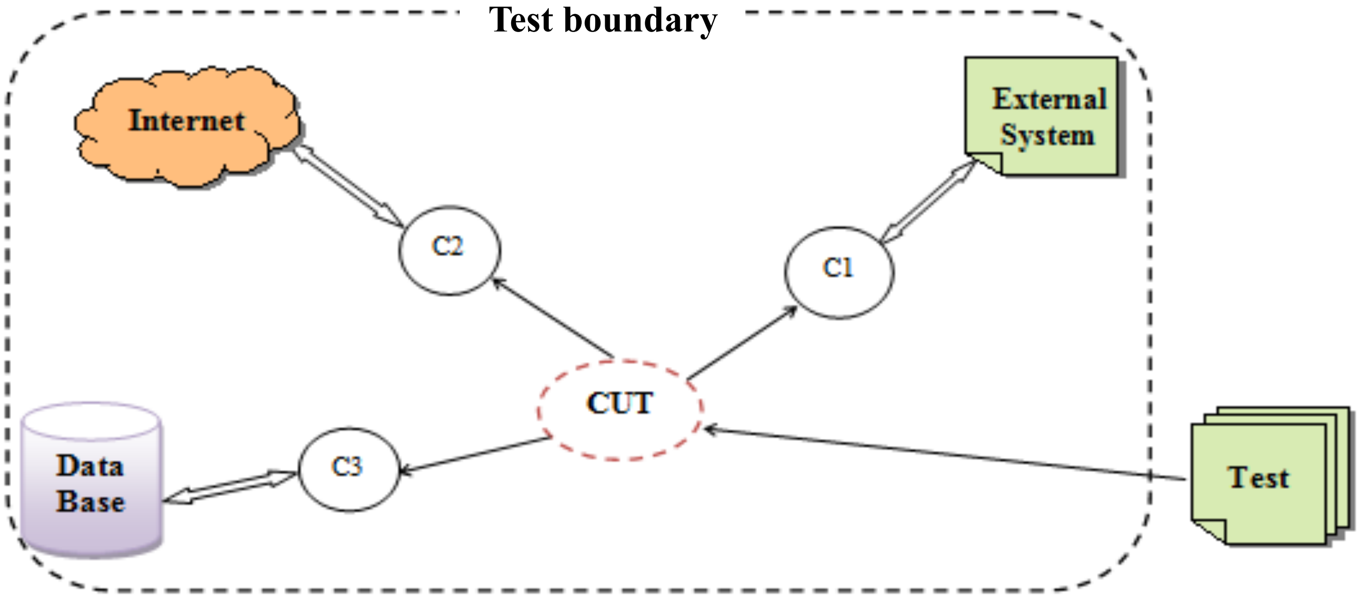
- Introduction
  - Unit Test
  - Mock Objects
- The Problem
- Solution
- Evaluation

Unit Test

- A unit is the smallest testable part of an application
- In the object oriented paradigm it is a class
- A unit test consists of a fixed sequence of method invocations with arguments that explores a particular aspect of the behavior of the class under test – CUT

Unit Test

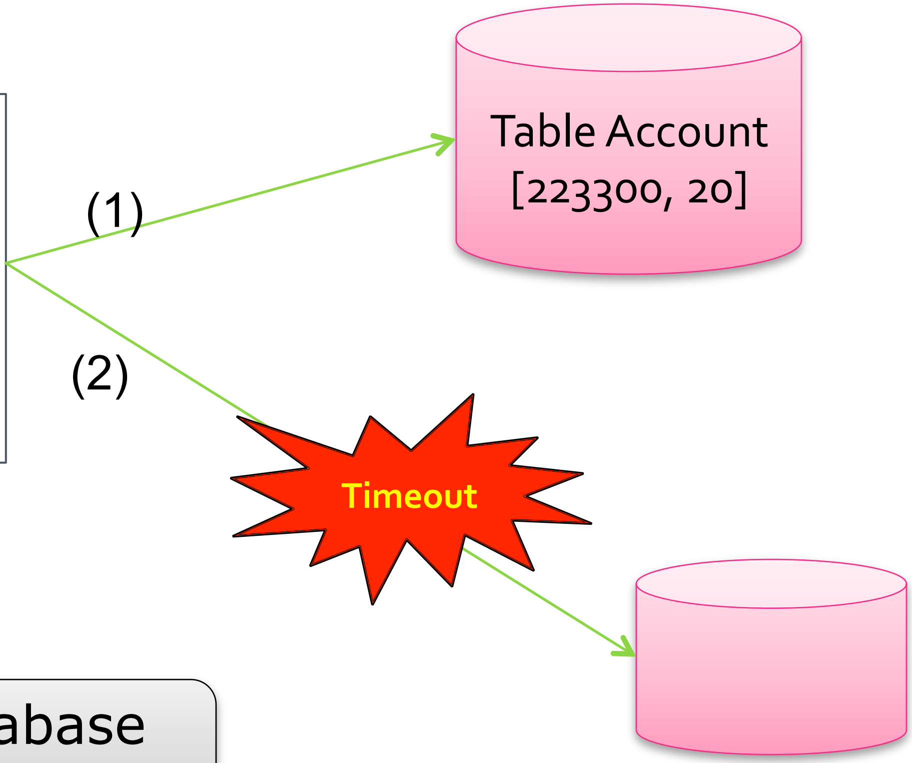
- Testing a unit in isolation is one of the most important principles of unit testing. However, the CUT usually depends on other classes…



An Example

- For example, we have a test that access data from a database

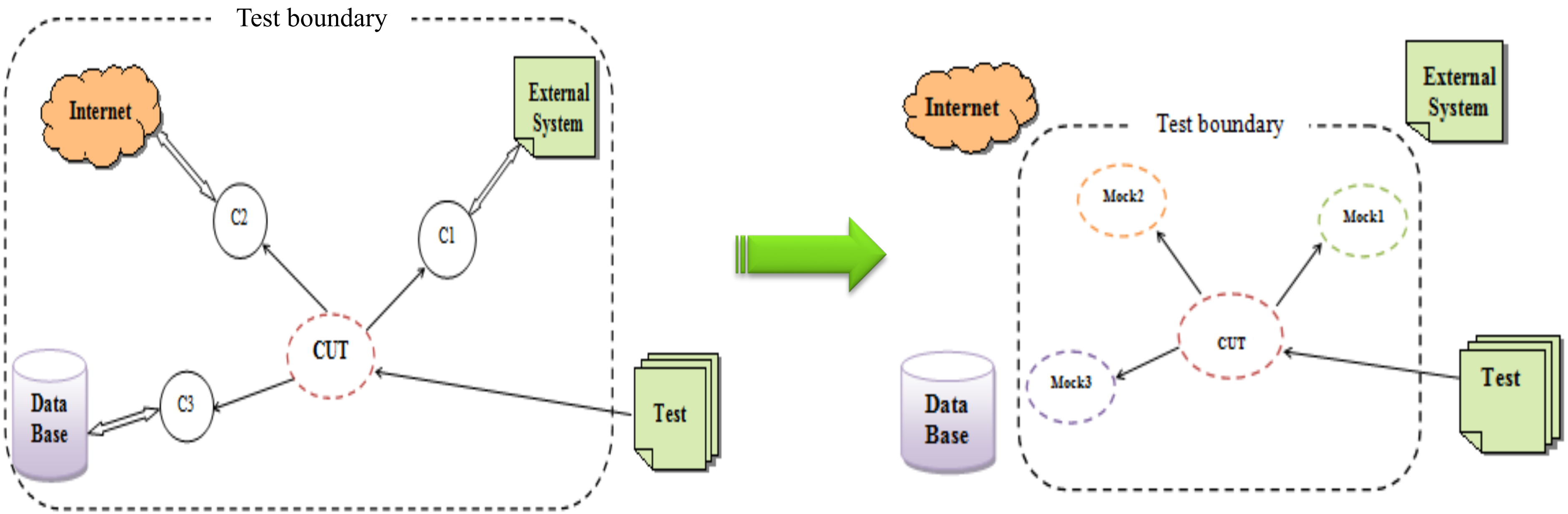
```
testMethodA(){

    ...
    insert("223300", 20);
    ...
}
```



It can be very slow!!!

Solution: Initiate the database with necessary data for the test

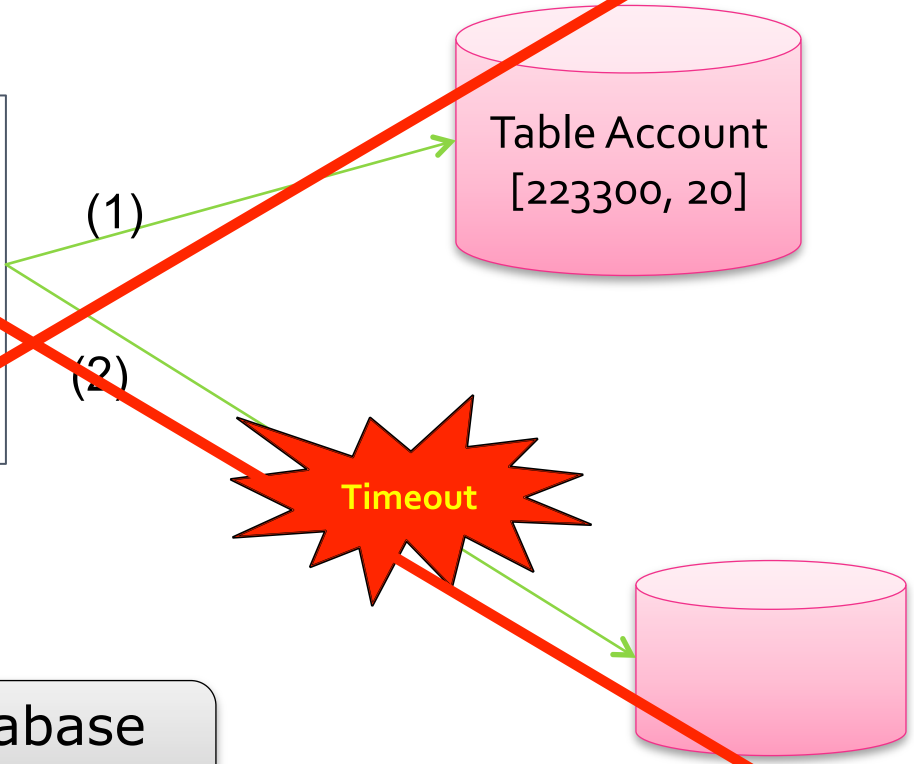
Testing with mock objects



An Example

- For example, we have tests that access data from a database

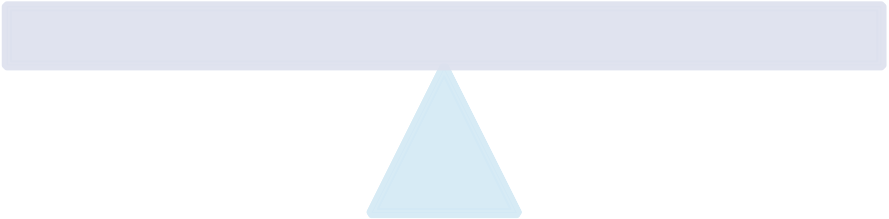
```
testMethodA(){

    ...
    insert("223300", 20);
    ...
}
```



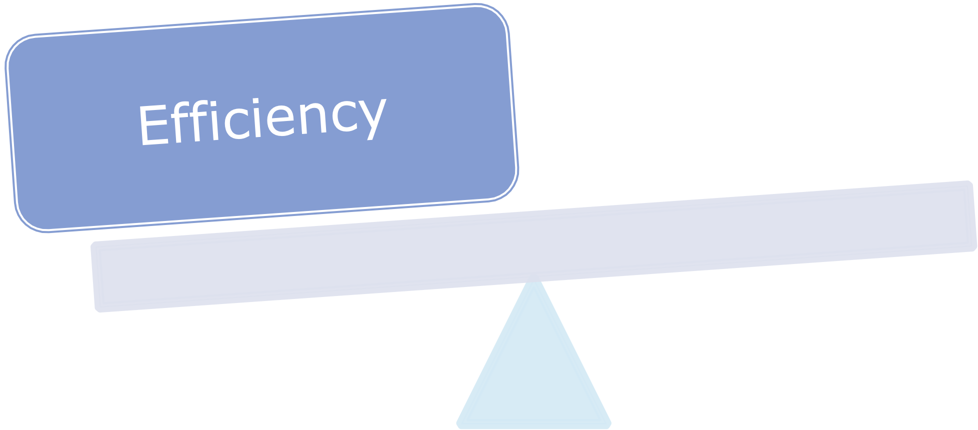
It can be very slow!!!

Solution: Initiate the database with necessary data for the test

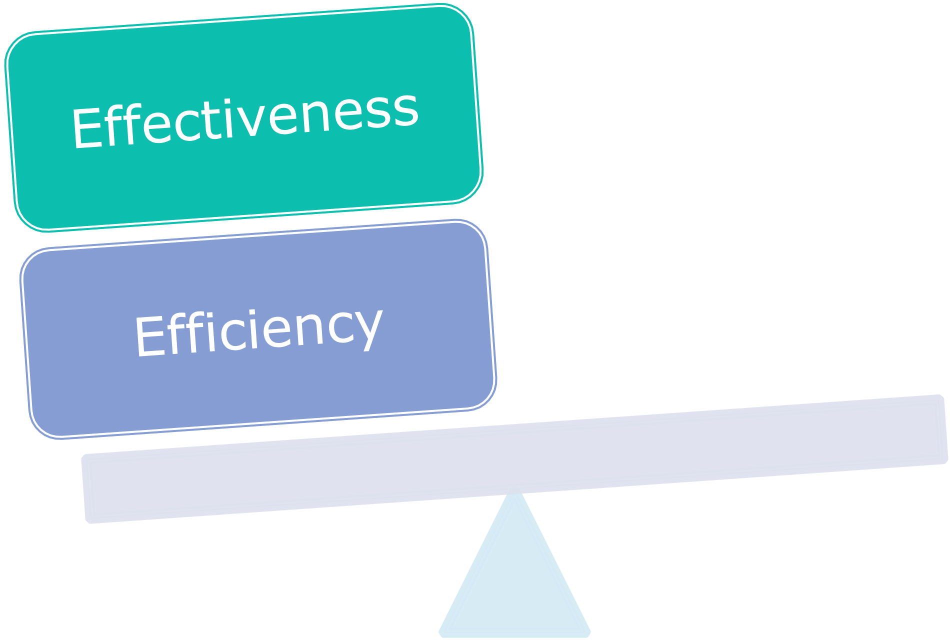
Mock Objects



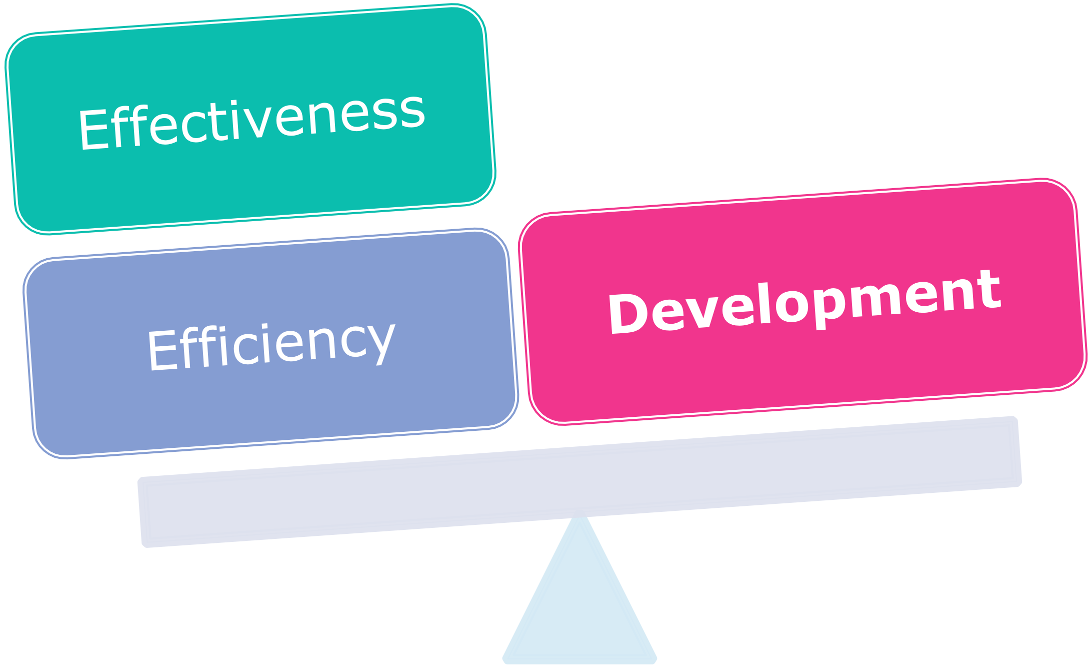
Mock Objects



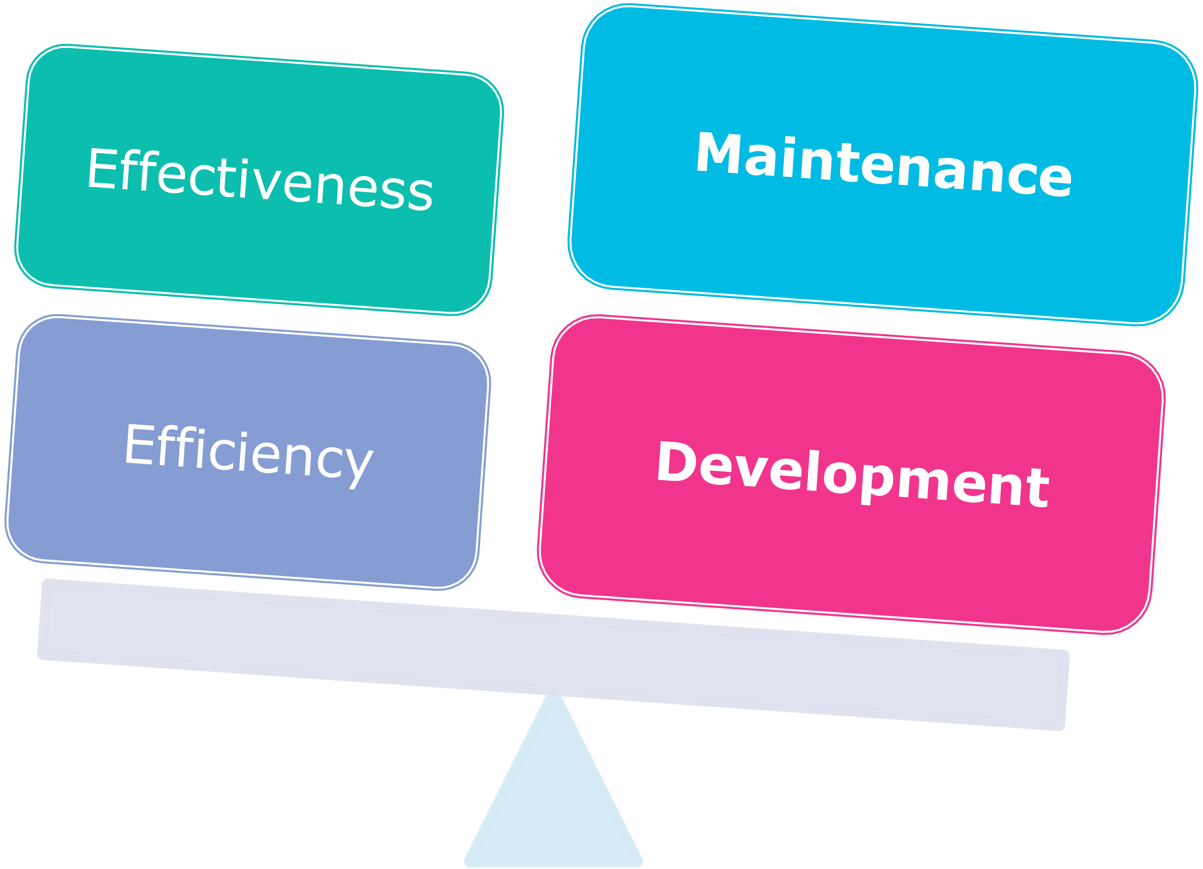
Mock Objects



Mock Objects



Mock Objects



Mock Objects

- Mocks can make tests more effective and efficient
  - Mocks make testing more effective because mock based tests provide more precise information about failures and defects
  - Mock based tests use simplified simulations of the real collaborators
  - They tend to run faster, especially when collaborators access databases, internet connections or other external systems

Mock Objects

- Writing tests that use mocks, however, can be a tedious and costly task
  - A mock based unit test consists in writing an expectation script
  - To be able to do so, testers must have a precise understanding of the interactions that occur among all objects in the chosen scenario
  - This includes not only the sequence of calls, but also the precise data that is sent and received back in each call
  - Mock-based tests are short-lived, they must be reconfigured for every minor design change of the objects in the scenario

Mock Objects

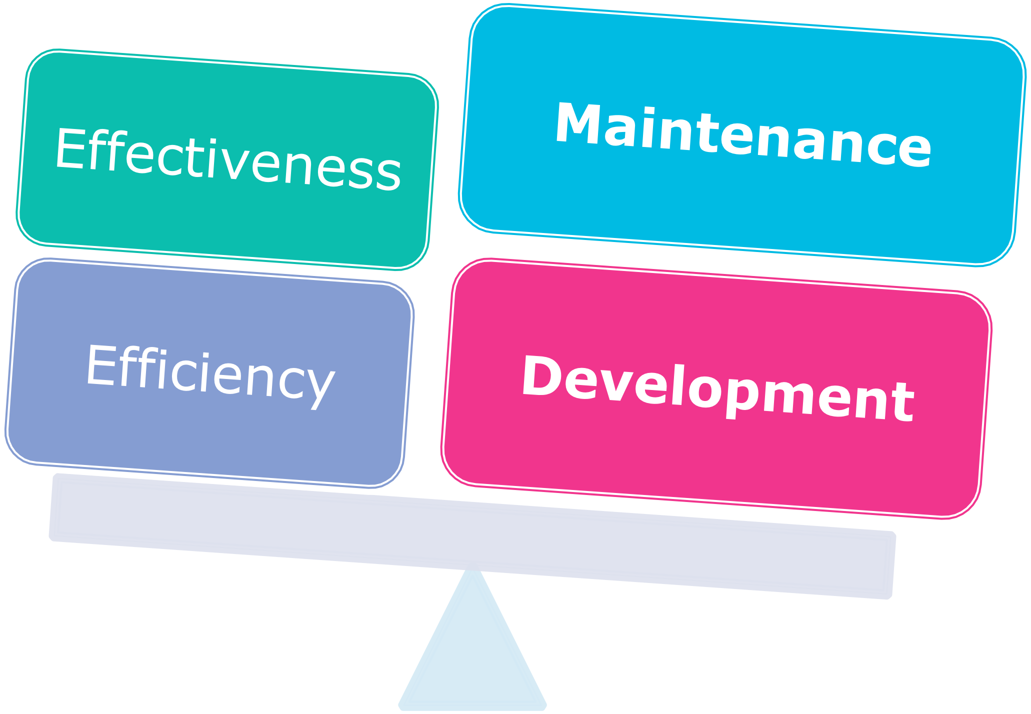
- All these drawbacks make using mocks costly
- As a consequence, in practice, testers must do cost-benefit analysis, and mock only part of the unit tests, when they don´t give up!

Mock Objects

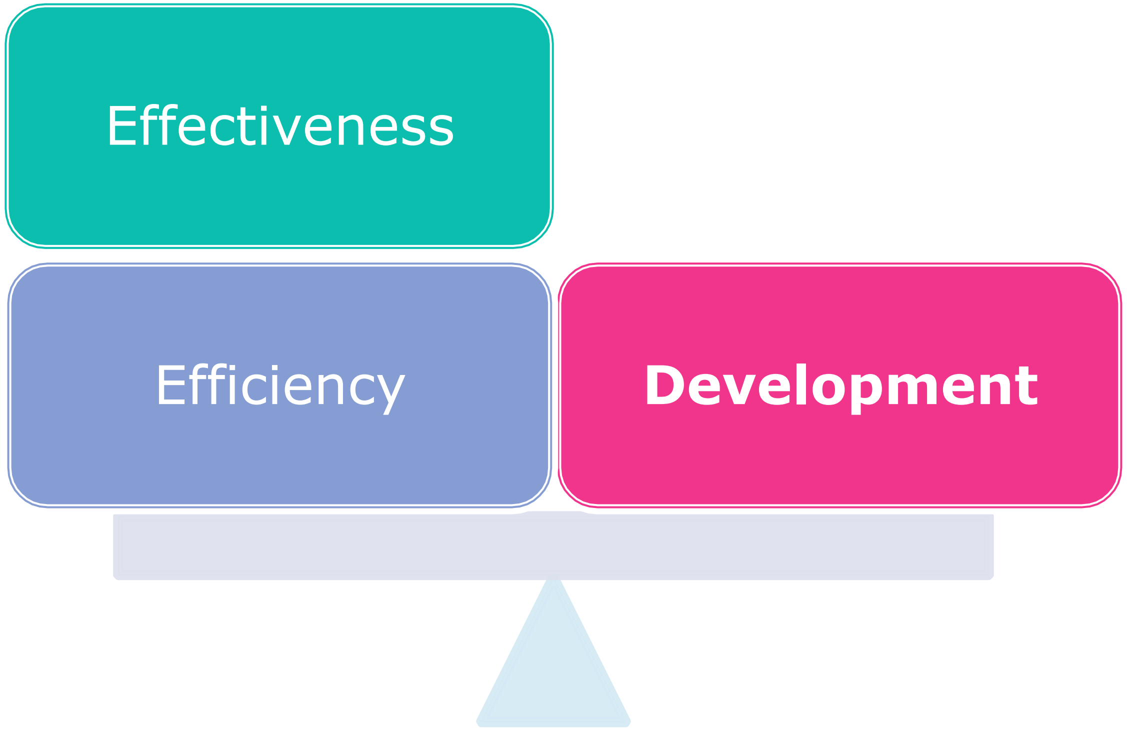
- In a test package, the percentage of unit tests with mock objects can be large
  - For instance, in OurBackup Home Project, 53.4% of all automatic unit tests use mock objects!

- Thus, how can we take advantage of mock objects without the burden caused by the development and maintenance?

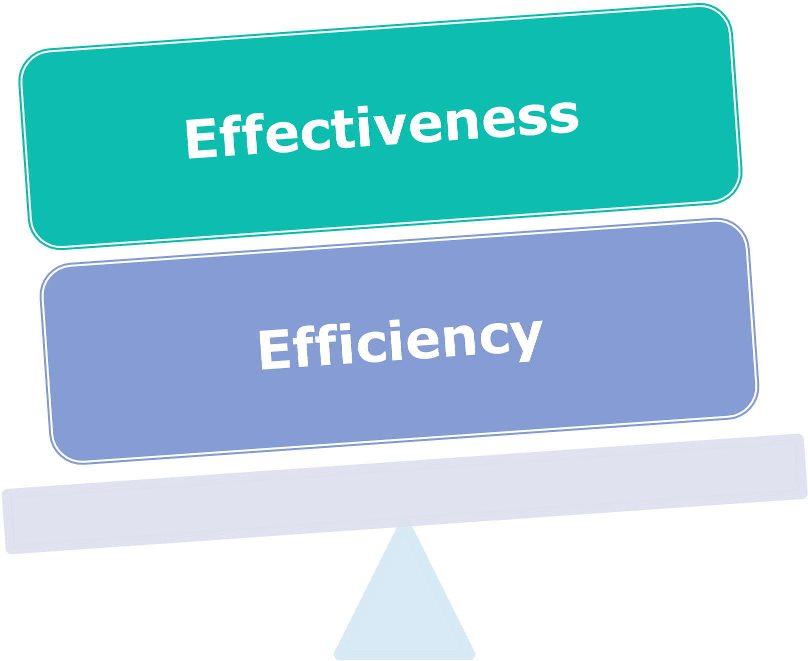
Solution



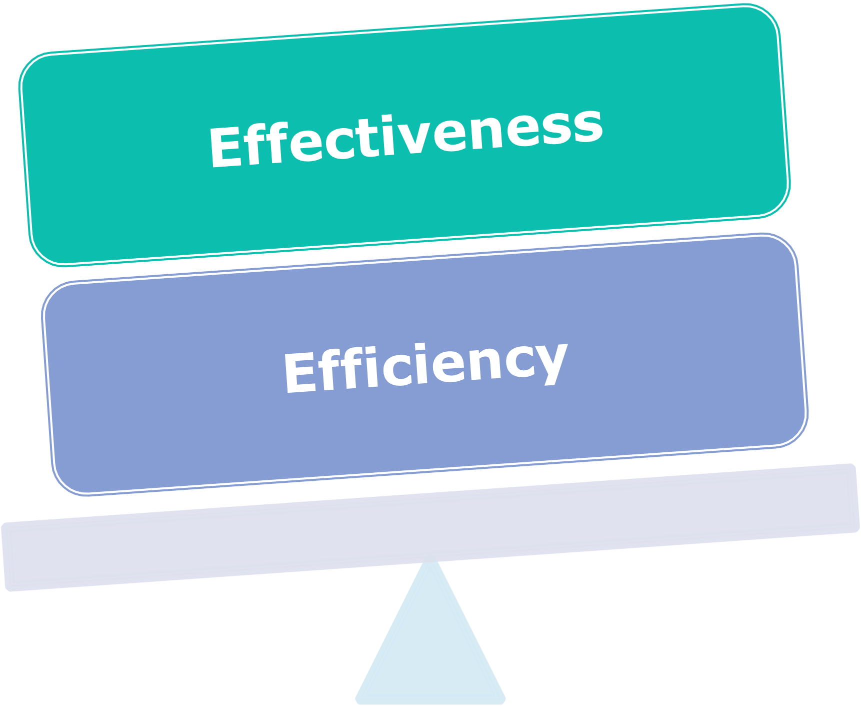
Solution



Solution



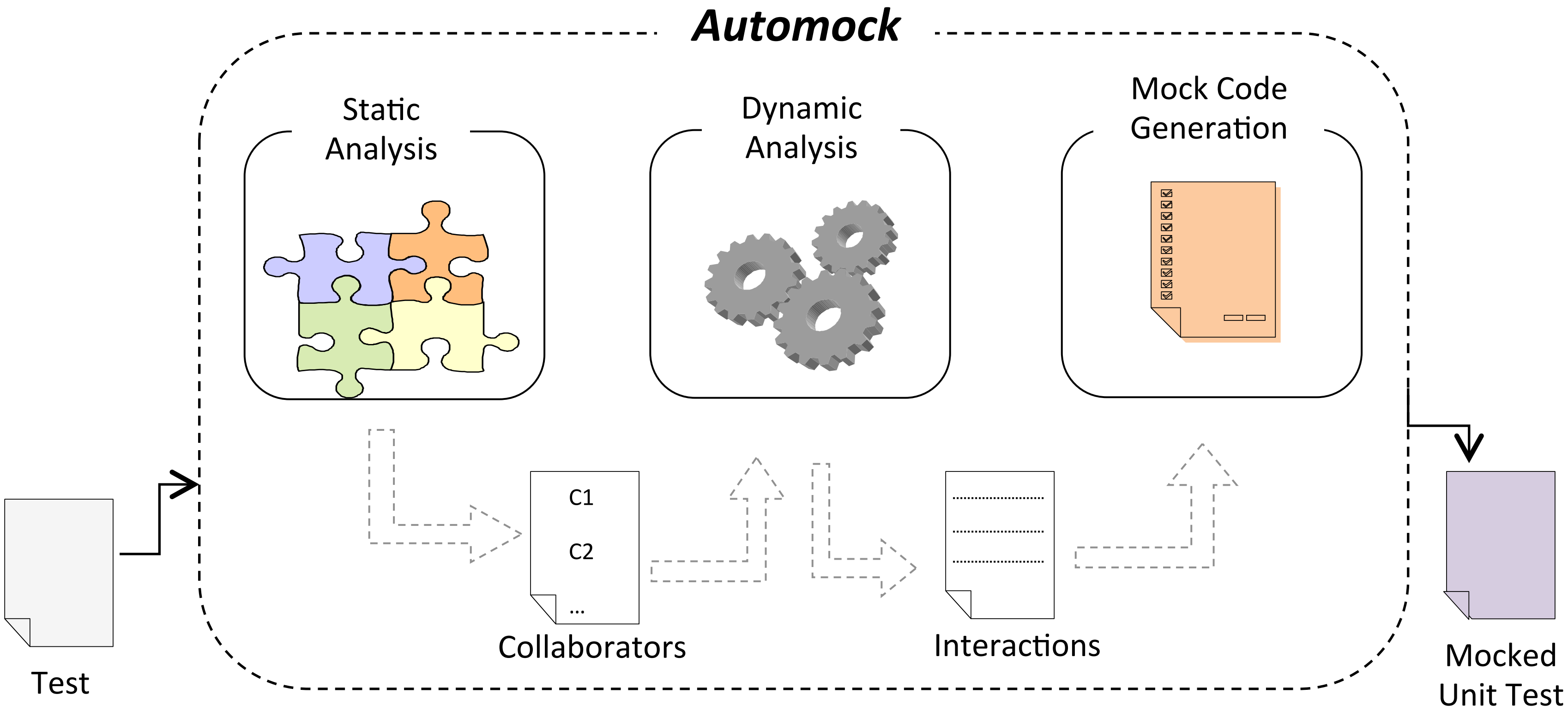
Solution



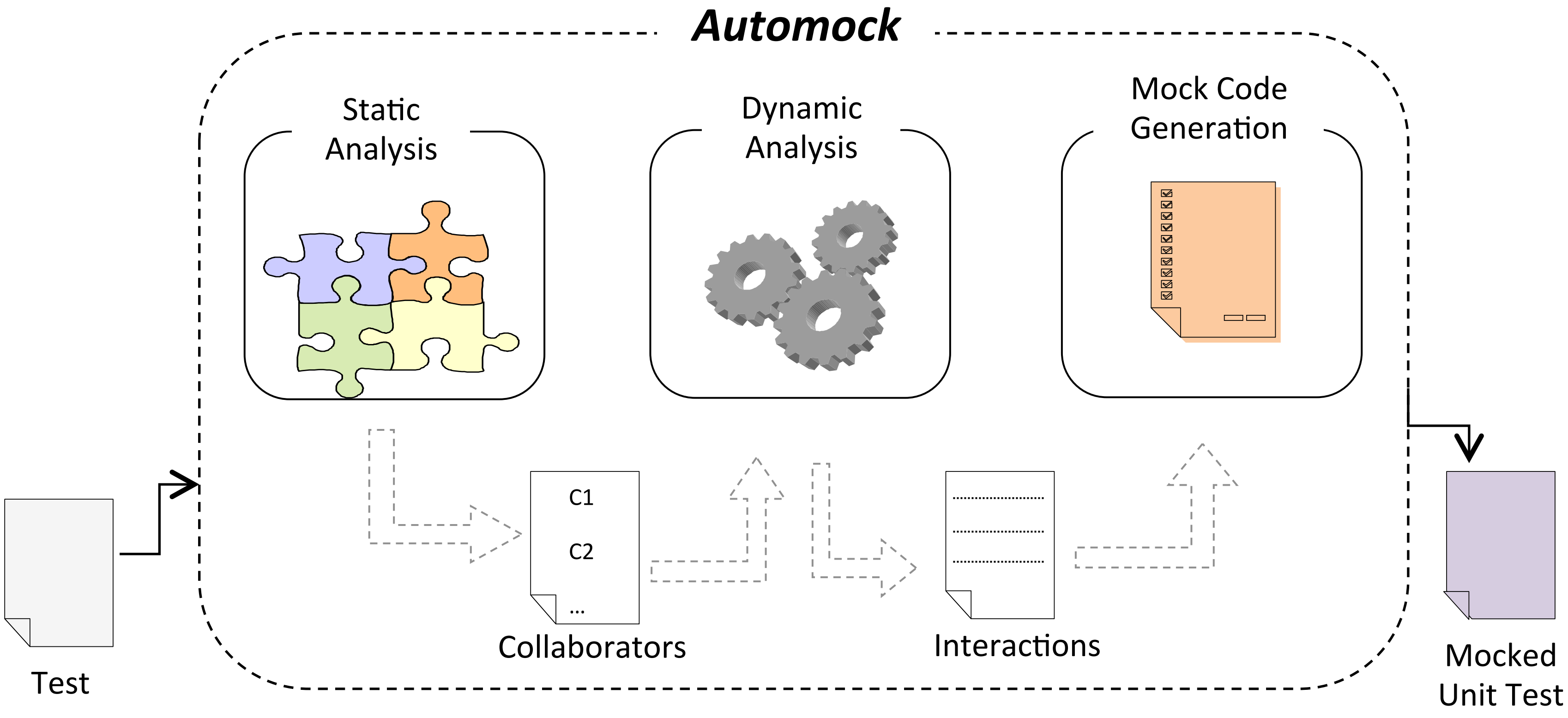
Automock

Automates the writing of mock objects, by generating them automatically for a given test class!

Solution: Automock



Solution: Automock



Preliminary Evaluation

- Questions:
  - What is the reduction of tester's effort, by using Automock?
  - What is the reduction of mock code development time, by using Automock?
  - Does the generated code have the same semantic of one developed manually?

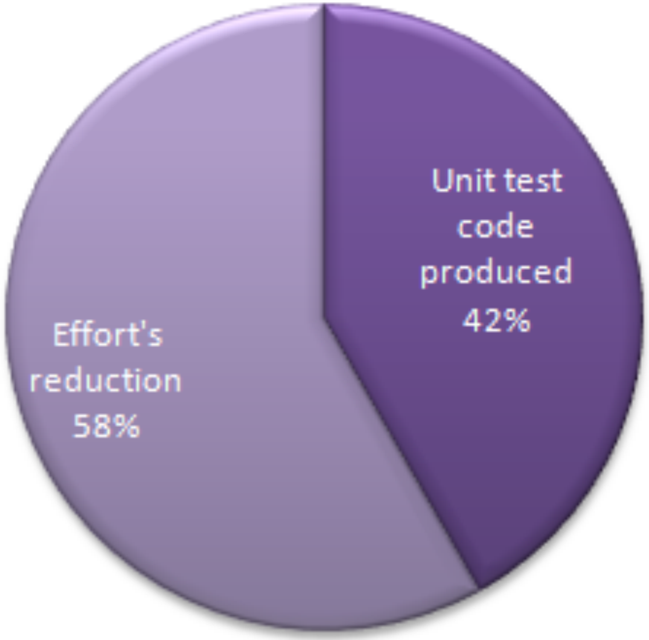
Preliminary Evaluation

GOAL	To investigate the tester's effort reduction on develop mock code, by comparing the number of lines of test code with and without mocks
QUESTION	What is the reduction of tester's effort, by using Automock?
METRIC	Effort's reduction = (LOTCM - LOTC) / LOTCM

Test Class	LOTC	LOTCM
Test1	243	600
Test2	905	2300

LOTC: lines of test code without mock objects
LOTCM: lines of test code with mock objects

It means that the tester/programmer will only have to produce the test code normally, and then generate mock code using *Automock*

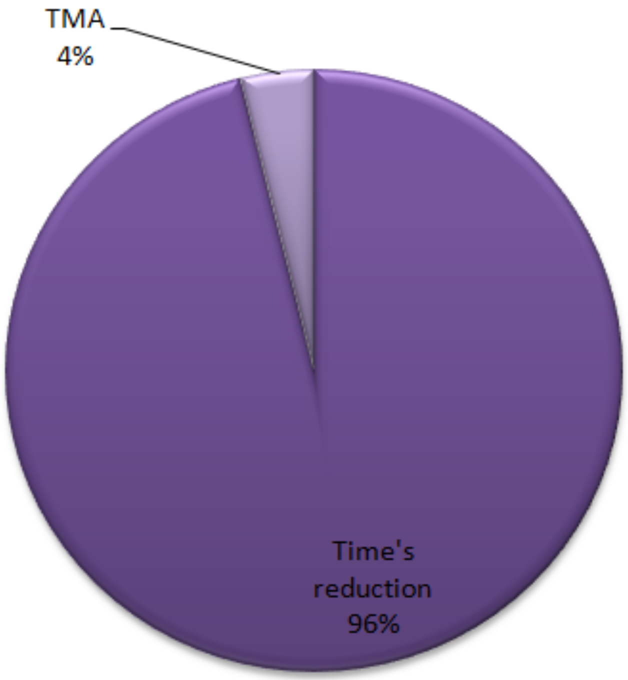


Preliminary Evaluation

GOAL	To examine the mock code development time through the comparison between develop mocks manually and automatically, by using Automock
QUESTION	What is the reduction of mock code development time, by using Automock?
METRIC	Time reduction = (TMM - TMA) / TMM

Test Class	TMM	TMA
Test1	30.33 min	15 min
Test2	6 hours	20 min

TMM: time to develop mock code manually
TMA: time to develop mock code automatically



It means that the tester/programmer will only take 4% of the time to produce mock code, by using *Automock*

Preliminary Evaluation

- Does the generated code have the same semantic of one developed manually?
  - To apply mutation tests in a manual mock code and in the one automatically generated
  - To do a qualitative analysis, by comparing manually the generated code and the one developed in a manual way
  - To execute the generated code and the one developed in a manual way with the purpose of obtain the same result

Conclusion an Future Work

- We presented a technique that automatically generates mock code for tests
- In order to support and evaluate the technique, we developed a prototype-tool and applied it in a test development environment
- Although the evaluation can only be considered as preliminary, the results are promising
- Furthermore, testers were convinced that the technique can be very helpful during test development

Conclusion an Future Work

- Future work will be focused in two directions:
  - First, we will further evaluate the gains that can be derived by applying the technique by means of more rigorous experiments
  - Second, we plan on evolving the tool to make it both more efficient and easier to use, possibly by developing an *Automock* plug-in to the Eclipse IDE

Thank you!

Questions? Suggestions?!