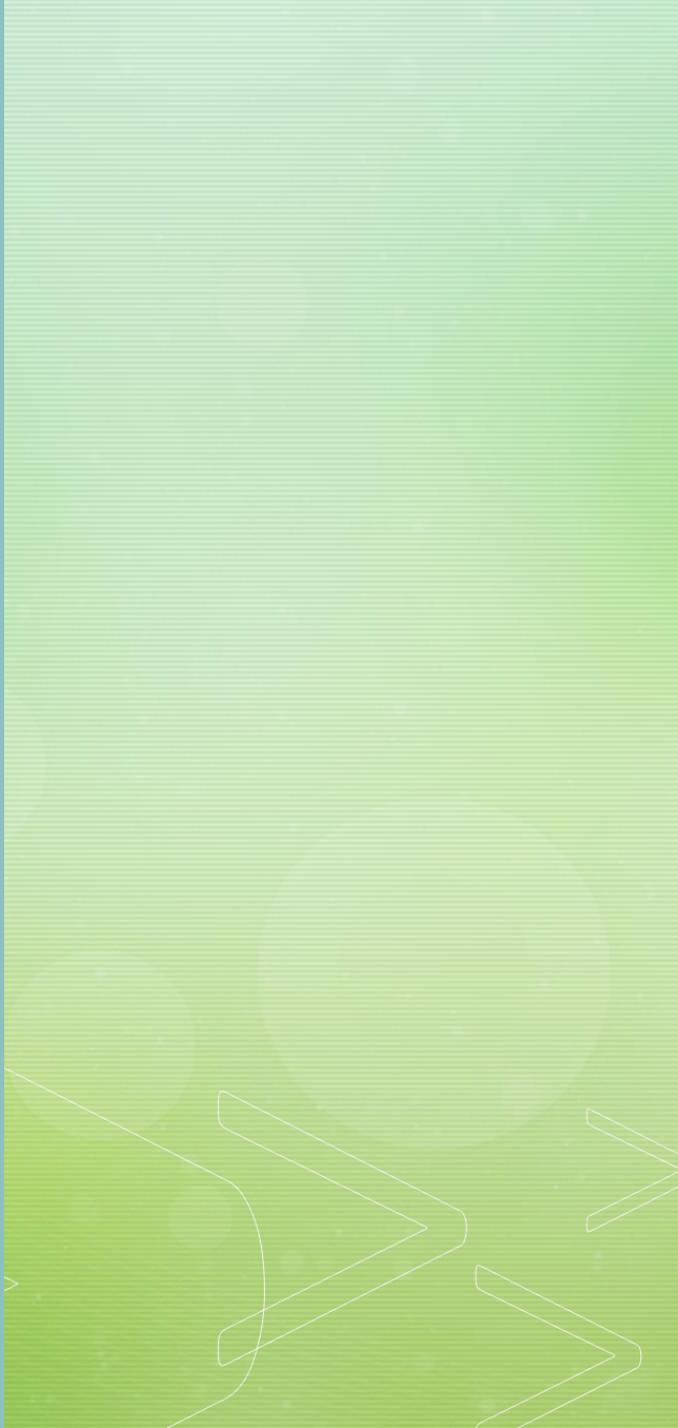


C++ User Group Aachen

Andreas Brack



Google Mock



Preface

- Previously own project, now absorbed into Google Test
- <https://github.com/google/googletest>
- License: BSD 3
- Linux: installable via package manager
 - Only header & sources, no ready to use libraries

Mock?

- A Mock object replaces the real object (in tests)
- Reasons for Mocking
 - Indeterministic data (Time, Temperature, etc)
 - Error cases are difficult to be produced (TCP-Errors)
 - Slow or complex operations
 - Unavailability (Other components, Interfaces, User Input)
 - Behavior of the real class (deleting something)

Limitations

- Only member functions of classes can be mocked
 - Build a class around free functions
- Design of the program / other classes has to support exchangeability of the class, which shall be mocked
- 2 Ways of getting mocks into a program
 - Inheritance (Interface)
 - Only virtual functions can be mocked
 - Derived class from real interface or pure virtual interface
 - Templates

Google Mock

- Helps defining mocks
- Main Features:
 - Call arguments check
 - Call times and Sequence check
 - Sideeffects and return value can be specified
 - Warn unexpected calls (default), StrictMock -> error, NiceMock -> default constructed return value
- Uses Macros for defining mocked functions
 - Offers scripting for support

Using Google Mock

- Create header only class
 - All functions to be mocked: Use macros to define Mock
 - `MOCK_METHOD2(foo, int(char, bool)) ; # Number of args, name, signature`
- Inside the Test: Define Behaviour

```
EXPECT_CALL(mock_object, method(matchers))  
    .With(multi_argument_matcher)? .Times(cardinality) ?  
    .InSequence(sequences) * .After(expectations) *  
    .WillOnce(action) * .WillRepeatedly(action)?  
    .RetiresOnSaturation(); ?
```

Example

- Application shall
 - Request User data (user input): Mocked
 - Do some operation (Dupilcate Vales): Tested
 - Store in some database: Mocked

Additional

- Many matchers are already defined
 - Pointers, Strings, containers, etc
- Extendable Matchers, Cardinalities, etc
- Template classes are mockable, `MOCK_* -> MOCK_*_T`
- Template member functions are not mockable
 - Workaround: Specialize for tested types and call mockable function inside specialization



Questions

