CMake

Tree

```
Project
|-- CMakeLists.txt
|-- docs
|   |-- CMakeLists.txt
|   |-- Doxyfile.in
|   |-- source
|       |-- conf.py
|       |-- index.rst
|-- environment.yml
|-- include
|   |-- library.h
|-- README.md
|-- src
|   |-- CMakeLists.txt
|   |-- some_source.cpp
|   |-- library.cpp
|-- tests
    |-- CMakeLists.txt
    |-- test_library.cpp
```

Root CMakeLists.txt

```
cmake_minimum_required(VERSION 3.20)
project(ProjectName)
set(CMAKE_CXX_STANDARD 17)
set(CMAKE_RUNTIME_OUTPUT_DIRECTORY
  ${CMAKE_BINARY_DIR}/bin)
set(CMAKE_LIBRARY_OUTPUT_DIRECTORY
  ${CMAKE_BINARY_DIR}/lib)
add_subdirectory(src)
add_subdirectory(docs)
enable_testing()
add_subdirectory(tests)
```

CMake: Sources

src/CMakeLists.txt

```
include_directories(${CMAKE_SOURCE_DIR}/include)
add_library(library SHARED library.cpp)
add_executable(some_binary some_source.cpp)
```

Tree

```
Project
|-- CMakeLists.txt
|-- include
|   |-- library.h
|-- src
    |-- CMakeLists.txt
    |-- some_source.cpp
    |-- library.cpp
```

CMake: Sources

library.h

```cpp
#pragma once
namespace library{
  /**
  * @brief Add two integers
  *
  * @param a First integer
  * @param b Second integer
  * @return int Sum of a and b
  */
  int add(int a, int b);
}
```

library.cpp

```cpp
#include "library.h"
namespace library{
  int add(int a, int b){
    return a + b;
  }
}
```

CMake: Tests

Tree

```
Project
|-- CMakeLists.txt
|-- include
|   |-- library.h
|-- tests
    |-- CMakeLists.txt
    |-- test_library.cpp
```

tests/test_library.cpp

```
#include <gtest/gtest.h>
#include "library.h"
TEST(LibraryTest, Test1){
  ASSERT_EQ(1, 1);
}
```

CMake: Tests

tests/CMakeLists.txt

```
include(FetchContent)
FetchContent_Declare(
  googletest
  GIT_REPOSITORY https://github.com/google/googletest.git
  GIT_TAG v1.15.2
)
FetchContent_MakeAvailable(googletest)
include(GoogleTest)
include_directories(${CMAKE_SOURCE_DIR}/src
  ${CMAKE_SOURCE_DIR}/include)
link_libraries(GTest::gtest_main)
add_executable(test_library test_library.cpp)
target_link_libraries(test_library library)
gtest_discover_tests(test_library)
```

CMake: Documentation

docs/CMakeLists.txt

```
configure_file(Doxyfile.in Doxyfile @ONLY) # Configure Doxyfile
doxygen # Generates ${DOXYGEN_OUTPUT_DIR}/xml/index.xml
# Generate build/docs/sphinx/html/index.html
sphinx-build -b html docs/source/conf.py -D
↪  breathe_projects.${PROJECT_NAME}=${DOXYGEN_OUTPUT_DIR}/xml
```

Doxyfile.in

```
Project
|-- docs
    |-- CMakeLists.txt
    |-- Doxyfile.in
    |-- source
        |-- conf.py
        |-- index.rst
```

```
# ...
INPUT = @DOXYGEN_INPUT_DIR@
OUTPUT_DIRECTORY = @DOXYGEN_OUTPUT_DIR@
# ...
```

CMake: Documentation

```
Project
|-- docs
    |-- CMakeLists.txt
    |-- Doxyfile.in
    |-- source
        |-- conf.py
        |-- index.rst
```

docs/source/conf.py

```
import os
project = os.getenv("PROJECT_NAME", "default name")
author = os.getenv("AUTHOR_NAME", "unknown")
copyright = os.getenv("COPYRIGHT", "2024, Raul")
extensions = ["breathe"]
breathe_default_project = project
breathe_default_members = ("members","undoc-members")
```

CMake: Documentation

```
Project
|-- docs
    |-- CMakeLists.txt
    |-- Doxyfile.in
    |-- source
        |-- conf.py
        |-- index.rst
```

docs/source/conf.py

```
Documentation
=====================================

Welcome to the landing page!
A section
----------

.. doxygennamespace:: library
```

CMake: Workflow

```
Project
|-- CMakeLists.txt
|-- include
|   |-- library.h
|-- src
|   |-- CMakeLists.txt
|   |-- some_source.cpp
|   |-- library.cpp
|-- tests
|   |-- CMakeLists.txt
|   |-- test_library.cpp
|-- docs
    |-- CMakeLists.txt
    |-- Doxyfile.in
    |-- source
        |-- conf.py
        |-- index.rst
```

```
mkdir build
cd build
cmake .. # In some cases -G "Ninja"
# Build libs, bins, tests and docs
cmake --build .
ctest # Run tests
# Open docs in a web-browser
open docs/sphinx/html/index.html
```

Troubleshooting: The softest skill

Failing forwards.

The expression problem

```
struct Expression{
  virtual double evaluate() const = 0;
};
class Constant: public Expression{
  double value;
  public:
  double evaluate() const override {
   ↪  return value; }
};
// ... 15 more Expressions ...
```

The expression problem

- Easy to add new expressions (classes)
- Hard to add new operations (methods)

The expression problem

```
double evaluate(Constant const& c){
  return c.value;
}
void print(Constant const& c){
  std::cout << c.value;
}
// ... 15 more operations ...
struct Constant{
  double value;
};
```

The expression problem

- Hard to add new expressions (classes)
- Easy to add new operations (methods)

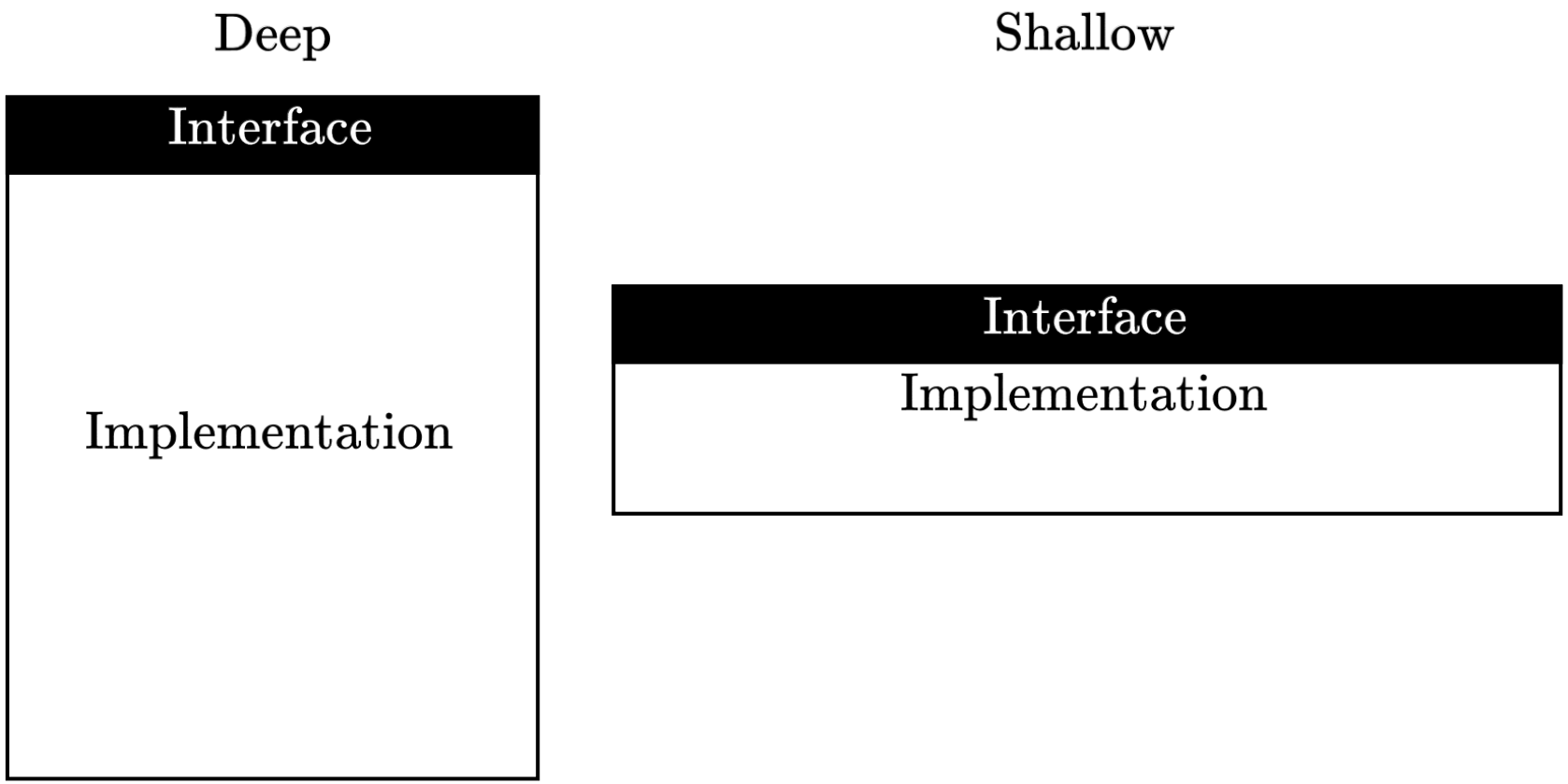
The expression problem and its solutions

Complexity

Anything related to the structure of a software system that makes it hard to understand and modify the system.

Reducing complexity: Cognitive load

- Deep vs. shallow modules



Deep vs. shallow modules

A deep module

```
class Interactor{
  virtual
  void sumForces(ParticleData &p) = 0;
};
```

Deep vs. shallow modules

A deep module

```
int open(const char *path, int flags, mode_t permissions);
ssize_t read(int fd, void *buf, size_t count);
ssize_t write(int fd, const void *buf, size_t count);
off_t lseek(int fd, off_t offset, int whence);
int close(int fd);
```

Encapsulated in `std::fstream`.

Deep vs. shallow modules

A shallow module

```
void appendValueToVector(std::vector<int> &v,
                         int value){
        v.push_back(value);
}
```

Hurtful dogmatic principles

SOLID

- Single Responsibility
- Open/Closed
- Liskov Substitution
- Interface Segregation
- Dependency Inversion

Classitist

- ~~Classes should be small~~

```
FileInputStream fileStream = new
 ↪  FileInputStream("file.txt");
BufferedInputStream bufferedStream = new
 ↪  BufferedInputStream(fileStream);
ObjectInputStream objectStream = new
 ↪  ObjectInputStream(bufferedStream);
```