Westfälische
Hochschule
Gelsenkirchen Bocholt Recklinghausen

Vorlesung „Grundlagen der Informatik und Programmierung 2“

Test-Driven-Development in C++

Einstieg

Prof. Dr. Tom Vierjahn

Visual Computing (https://vc.w-hs.de)
Fachbereich Wirtschaft und Informationstechnik – Campus Bocholt

Sommersemester 2021

Unit Test[1]

Westfälische Hochschule

„A unit test verifies the behavior of a code unit,
where a unit is the smallest testable piece of an application.“

Ablauf:

1. Kontext einrichten (optional)
2. zu testendes Verhalten ausführen
3. Ergebnis überprüfen
4. aufräumen (optional)

[1] Jeff Langr: *Modern C++ Programming with Test-Driven-Development*. Pragmatic Bookshelf, 2013. ISBN 9781937785482.

Test-Driven Development[2]

Westfälische Hochschule

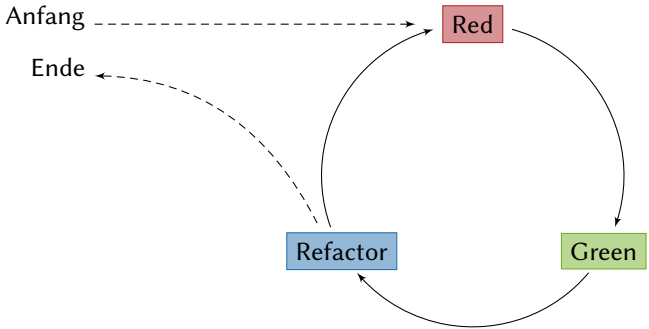
„Test-driven development is a set of techniques that any software engineer can follow, which encourages simple designs and test suites that inspire confidence.“

Regeln:

1. Schreibe erst fehlschlagenden, automatisierten Test; *erst danach* Code.
2. Entferne Verdoppelungen.

[2] Kent Beck: *Test Driven Development: By Example.* Addison-Wesley Professional, 2002. ISBN 9780321146533.

Der TDD-Cycle



Compiler

Westfälische Hochschule

„In computing, a compiler is a computer program that translates computer code written in one programming language (the source language) into another language (the target language).“[3]

Was machen wir daraus?

- Compiler verarbeitet gesamten Quellcode.
- Compiler *beurteilt* gesamten Quellcode.

[3] W https://en.wikipedia.org/wiki/Compiler (30. April 2021)

Compiler-Flags

Westfälische Hochschule

- Compiler wird durch *Compiler-Flags* „gesteuert“

z. B. GCC[4] (gelten auch für LLVM Clang[5]):

- `-Wall`:
  „This enables all the warnings about constructions that some users consider questionable, and that are easy to avoid …“
- `-Wextra`:
  „This enables some extra warning flags that are not enabled by `-Wall`.“
- `-Wpedantic`:
  „Issue all the warnings demanded by strict ISO C and ISO C++ …“
- `-Werror`:
  „Make all warnings into errors.“

[4] https://gcc.gnu.org (30. April 2021)
[5] https://clang.llvm.org

Compiler-Flags

Westfälische Hochschule

- Compiler wird durch *Compiler-Flags* „gesteuert“

z. B. MSVC[6]:

- `/W3`:
  „displays [...] (production quality) warnings ...“
- `/W4`:
  „displays level 1, level 2, and level 3 warnings, and all level 4 (informational) warnings that aren't off by default.“
- `/WX`:
  „Treats all compiler warnings as errors.“
- `/Wall`:
  „Displays all warnings displayed by `/W4`
  and all other warnings that `/W4` doesn't include –
  for example, warnings that are off by default.“

[6] https://docs.microsoft.com/en-us/cpp/build/reference/compiler-option-warning-level (30.04.2021)

Style-Guide

Westfälische Hochschule

viele Möglichkeiten:

- Google C++ Style Guide[7]
- C++ Core Guidelines[8]
- LLVM Coding Standards[9]
- …

wichtig:

- hängt vom Team und der Umgebung ab
- besser nicht selber schreiben und warten
- sollte automatische Kontrolle und Formatierung bieten

[7] https://google.github.io/styleguide/cppguide.html
[8] https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines
[9] https://llvm.org/docs/CodingStandards.html

cpplint[11]

static code checker for C++

Westfälische Hochschule

- prüft, ob Google-C++-Styleguide befolgt wird

Installation:

- macOS mit Homebrew[10]: `brew install cpplint`
- ansonsten: `pip install cpplint`

CMake-Integration (ab CMake 3.8):

- über Variable `CMAKE_C_CPPLINT` oder `CMAKE_CXX_CPPLINT`
- hat allerdings Grenzen

[10] https://brew.sh
[11] https://github.com/cpplint/cpplint

ClangFormat[12]

Westfälische Hochschule

- formatiert unter anderem C++-Code

Installation:

- über Paketmanager oder Installer;
  entweder direkt oder als Teil des Clang-Pakets
- als IDE-Plugin

Wichtig:

- für Editor-Integration sorgen

[12] https://clang.llvm.org/docs/ClangFormat.html

The Pitchfork Layout (PFL)[13]

vorgeschlagen von Colby Pike

Westfälische Hochschule

gebaute Produkte:

- `build/`: nicht Teil der Projekt-Quellen, muss ignoriert werden

Projektquellen:

- `src/`: zu kompilierender Quelltext
- `include/`: öffentliche Header
- `tests/`: Unit-Tests

identische Unterverzeichnisstruktur entsprechend einzelner Komponenten

Module – bei größeren Projekten:

- `bin/`: Anwendungs-Module
- `lib/`: Bibliotheks-Module

Unterverzeichnisse: Module; darunter jeweils siehe *Projektquellen*

[13] https://github.com/vector-of-bool/pitchfork

The Missing Package Manager / Dependency Manager

Westfälische Hochschule

C++ bringt keinen Paketmanager mit.

Warum eigentlich nicht?

Conan[14] – „the C/C++ Package Manager“:

- open source
- dezentral
- multi-plaform

Conan-Center[15]:

- ein mögliches Paket-Repository

[14] https://conan.io
[15] https://conan.io/center/

Unit-Testing-Framework

Westfälische Hochschule

Catch2[16]:

- open source
- selbst-registrierende Tests

Alternativen:

- GoogleTest[17]
- Boost.Test[18]
- CppUnit[19]
- doctest[20]

[16] https://github.com/catchorg/Catch2
[17] https://github.com/google/googletest
[18] https://www.boost.org/doc/libs/1_66_0/libs/test/doc/html/index.html
[19] https://sourceforge.net/projects/cppunit/
[20] https://github.com/onqtam/doctest

Zusammenfassung

Westfälische Hochschule

- Unit-Tests
- Test-Driven Development – Einführung
- Compiler und -Flags
- Styleguide
- Projektlayout
- Paketmanager
- Unit-Testing-Frameworks

Prof. Dr. Tom Vierjahn

- tom.vierjahn@w-hs.de

Visual Computing

- https://vc.w-hs.de
- VisualComputingWH
- Visual Computing WH
- @VisComputingWH

Westfälische Hochschule
Fachbereich Wirtschaft und Informationstechnik
Campus Bocholt