Prof. Dr. Ralf Hahn
Prof. Dr. Urs Andelfinger
Prof. Dr. Stefan T. Ruehl
Michael Guist
Lukas Köhler

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

# Praktikum Software Engineering

## Allgemeines

Im Rahmen des Software Engineering-Praktikums sollen Sie verschiedene Methoden und Techniken, die in der Vorlesung behandelt werden, praktisch anwenden und vertiefen. Dazu wird der Cocktail-Mischautomat "CocktailPro" aus dem Jahr 2015 nun unter den Randbedingung eines professionellen Projektes mit Auflagen bezüglich Qualität und Wartbarkeit weiterentwickelt.

## Vorgehen

Nachfolgend ist die Zuordnung der Praktikumstermine zu den einzelnen Projektaktivitäten und Aufgaben dargestellt.

| **Termin und Thema** | **Projektaktivitäten / Aufgaben für Projektteam** |
|---|---|
| 1. Kickoff und Gruppenbildung | Ihr Auftraggeber erläutert Ihnen die Ziele des Praktikums und des Projekts "CocktailPro_2022". Es erfolgt die Einteilung in 4er Gruppen für das Praktikum sowie ein Überblick über Git.<br>**Nachbereitung:** Installation von git und der IDE CLion sowie Übungsaufgaben zum lokalen Arbeiten mit git. (Andere IDEs sind möglich, aber nur für CLion können wir – eng begrenzten - Support bieten). |
| 2. Git-Anbindung an das zentrale Repository und Codeübergabe für Ihre IDE. | Es wird die Versionsverwaltung mit Git aufgesetzt. Sie erhalten von Ihrem Auftraggeber den aktuellen Stand der Software CocktailPro und Sie nehmen die Anbindung der IDE vor. Sie üben den Arbeitszyklus im Team.<br>**Nachbereitung:** Kommentieren des Quellcodes und Arbeit mit Git. |
| 3. Automatischer Build mit Jenkins | Wir richten Ihren Zugang zur Jenkins-Oberfläche mit automatischem Build und diversen anderen Qualitätsüberprüfungen ein. Dabei erzeugen Sie eine Dokumentation aus den von Ihnen erstellten Kommentaren mit Doxygen und führen eine statische Codeanalyse mit CppCheck durch.<br>**Nachbereitung:** Korrektur der Befunde von CppCheck |
| 4. Metriken und Code-Qualität | Sie vermessen den Code mit cccc und einigen Standard-Metriken. Durch Refactorings werden verdächtige Codestellen qualitativ verbessert.<br>**Nachbereitung:** Weitere Refactorings zur Verbesserung der Codequalität |
| 5. Systematisches Testen | Durch den Einsatz von GoogleTest kann C++-Code automatisch getestet werden. Mit Hilfe der Testüberdeckung ("Coverage") erkennen Sie Codeteile, die noch ungetestet sind und ergänzen entsprechende Tests.<br>**Nachbereitung:** Weitere Tests zur Erhöhung der Test Coverage |
| 6. Meilenstein und Reflektion | Der ursprüngliche Code wurde bisher mit diversen Verfahren untersucht und die Qualität durch entsprechende Maßnahmen verbessert. Der Code stellt jetzt die Basis für die Weiterentwicklung dar. Die durchgeführten Qualitätsmaßnahmen werden diskutiert und ergänzt.<br>**Nachbereitung:** Weitere Tests und andere Maßnahmen nach Bedarf |

| | |
|---|---|
| 7. Weiterentwicklung mit Qualität | Ihr Auftraggeber stellt Kundenwünsche vor, die Sie umsetzen sollen. Sie erfassen die Wünsche als User Stories und gehen die Umsetzung unter Erhaltung der bisherigen Qualitätsmaßstäbe an.<br>**Nachbereitung:** Umsetzung der User Stories |
| 8. Planung & Entwicklung - die 1. Runde | Ihr Auftraggeber stellt weitere Funktionen vor, die Sie umsetzen sollen. Dieses Mal sollen Sie aber eine Prognose abgeben, wie lange das dauern wird. Sie schätzen die Aufwände einzeln und im Team.<br>**Nachbereitung:** Umsetzung der Stories mit Erfassung des tatsächlichen Aufwands |
| 9. Planung & Entwicklung - die 2. Runde | Wir diskutieren die Unterschiede zwischen den Aufwandsschätzungen und Ihrem tatsächlichen Aufwand und suchen Ursachen dafür. Dann stellt Ihr Auftraggeber eine neue Story vor und Sie versuchen mit den neuen Erkenntnissen eine bessere Schätzung.<br>**Nachbereitung:** Umsetzung der Story mit Erfassung des tatsächlichen Aufwands |
| 10. Scrum, Continuous Integration und vieles mehr | Durch die vorgegebenen Methoden und die Entwicklung der Stories haben Sie viele Techniken und Herausforderungen in der Software-Entwicklung kennengelernt. Wir diskutieren im Zusammenhang der Erfahrungen diverse Begriffe aus der agilen Entwicklung.<br>**Nachbereitung:** Erstellen einer Präsentation zu vorgegebenen Themen |
| 11. Projektab-schluss und Bewertung | Sie präsentieren Ihre Erfahrungen in Bezug auf Themen wie Teamarbeit, Legacy Software, Qualitätsmaßnahmen uvm.<br>Offene Fragen im Umfeld des Software Engineering können diskutiert werden. |

## Organisatorisches

Das Praktikum erstreckt sich auf 11 Termine pro Gruppe. **Es besteht Anwesenheitspflicht im Praktikum.** Die Praktikumsaufgaben werden zwischen den Terminen auf Ihren Privatrechnern bearbeitet. Die Abnahme erfolgt auf den bereitgestellten Servern mit den vorgegebenen Werkzeugen.

Jede Praktikumsaufgabe beinhaltet einen Nachbereitungsteil, der deutlich vor dem nächsten Praktikumstermin bearbeitet werden muss.

## Software Engineering ist anders!

In Software Engineering ist die Software-Entwicklung nur das Mittel zum Zweck. Es geht weniger darum WAS entwickelt wird, sondern eher darum WIE Sie die Software entwickeln. Der Fokus liegt auf Techniken zur Steigerung der Qualität und Zuverlässigkeit.

Sie entwickeln als ein Team und die Koordination des Teams ist ein wichtiger Bestandteil der Aufgabe. Die Praktikumstermine dienen vorwiegend der Koordination des Teams und dem Kennenlernen von

neuen Techniken. Die Anwendung der Techniken erfolgt dann in der Nachbereitung. Planen Sie ca. 3 Stunden[1] pro Person und Praktikumstermin für diese Nachbereitung ein.

## Abgabeverfahren

Damit Ihr Auftraggeber sich den Arbeitsfortschritt anschauen kann, muss die Abgabe deutlich vor dem nächsten Treffen (Praktikumstermin) erfolgen. **Deshalb müssen Ihre Arbeitsergebnisse immer 48h vor dem nächsten Praktikumstermin im Versionsverwaltungssystem (Git) veröffentlicht werden. Die Bewertung Ihrer Abgabe bezieht sich auf diesen Stand - auch wenn danach noch Änderungen gemacht werden!**

Durch manuelle Inspektion oder mit entsprechenden Tools wird überprüft, ob Sie das Ziel der Übung erreicht haben. Jedes Teammitglied ist in der Lage, seine Tätigkeit für den aktuellen Praktikumstermin zu präsentieren. **Dabei ist für jedes Teammitglied im persönlichen git-Branch nachzuweisen, was Sie jeweils individuell seit dem letzten Termin neu entwickelt oder beigetragen haben.**
Wenn ein Team die geforderte Aufgabe nicht erledigt hat, gibt es eine Verwarnung für das Team.

## Teamarbeit

Sie arbeiten in der Regel in einer 4-er Gruppe als Team. Die selbständige Planung, Koordination und termingerechte Erfüllung der Aufgaben des Teams ist ein wichtiger Bestandteil des Praktikums. Die vollständige Erfüllung der Aufgaben liegt in der Verantwortung des Teams. Deshalb gelten Verwarnungen immer für alle Teammitglieder. Falls es Probleme mit einem Teammitglied gibt, die Sie nicht im Team lösen können, informieren Sie bitte möglichst frühzeitig Ihren Betreuer!

Es wird vorausgesetzt, dass jedes Teammitglied sich bei jedem Termin aktiv beteiligt. Und zwar so, dass jedes Teammitglied auch jedes Thema bearbeitet (Eine Aufteilung der Art "Person X macht die Kommentare, Person Y macht die Implementierung" ist NICHT zulässig).
**Durch das Arbeiten mit Versionskontrolle (Git) wird ständig protokolliert, wer wann welche Änderung gemacht hat[2]. Dadurch wird schnell ersichtlich, wer woran gearbeitet hat und wer nicht.** Insbesondere ist auch bei einer Erkrankung oder Verspätung am Praktikumstermin prüfbar, ob Sie Ihren Teil der Arbeit bereits erbracht haben oder nicht.

## Zentrale Abnahmeumgebung

Nachfolgend ist die Software dargestellt, die in diesem Software Engineering Praktikum auf unserer Abnahmeumgebung eingesetzt wird. (Wenn Sie die von uns empfohlene CLion-IDE verwenden, ist sichergestellt, dass Ihre lokale Umgebung dazu kompatibel ist! Falls Sie andere Umgebungen benutzen, müssen Sie sich selbst darum kümmern.).

### C++ Compiler

Die Entwicklung der Anwendung erfolgt in C++ 11. Wir verwenden den Gnu Compiler (https://gcc.gnu.org/ ) in der aktuellen Version. Unter Windows wird die Verwendung von Cygwin (https://www.cygwin.com/) mit g++, make, gdb und cmake empfohlen.

[1] Dabei wird vorausgesetzt, dass Sie bereits gut Programmieren können und sicher mit einer IDE umgehen können. Wenn das nicht der Fall ist, werden Sie mehr Zeit für das Praktikum brauchen.
[2] Deshalb sollten Sie auch darauf achten, dass Sie Ihre Arbeitsergebnisse selbst im Git hochladen.

### GoogleTest

Wir verwenden GoogleTest (https://github.com/google/googletest ) als Framework zur Durchführung von Unit-Tests. (CLion verwendet das google-Test Framework automatisch. Sie brauchen sich nicht um die Installation kümmern).

### IDE (CLion)

Die Verwendung einer professionellen IDE wird an Hand von CLion (https://www.jetbrains.com/clion) erläutert. Eine Lizenz können Sie hierfür kostenlos direkt vom Hersteller erhalten; Details hierzu finden Sie weiter unten.

### Git - Konfigurationsmanagement-Werkzeug

Um die verschiedenen Entwicklungsstände zu versionieren und ein paralleles Arbeiten zu ermöglichen, wird die Software git (https://git-scm.com) eingesetzt. Das zentrale Repository befindet sich auf einem Server, des Fachbereichs (https://code.fbi.h-da.de/).

### Jenkins - Continuous Integration-Server

Auf einem Build-Server erfolgt der automatische Build der Anwendung mit Jenkins (https://jenkins.io/) und diversen Plugins. Ein Jenkins-Server wird auf einem Server des Fachbereichs bereitgestellt. Jenkins ermöglicht das automatische Beschaffen und Kompilieren des aktuellen Quellcodes aus dem Konfigurationsmanagement. Anschließend werden automatisierte Tests durchgeführt und diverse Qualitätsprüfungen durchgeführt.

### Doxygen

Aus den Kommentaren im Code kann eine übersichtliche Dokumentation extrahiert werden, wenn Sie eine bestimmte Notation berücksichtigen. Während des Jenkins-Builds erzeugt Doxygen (www.doxygen.org/ ) eine Dokumentation im HTML-Format.

### lcov

Beim Ausführen der Tests kann automatisch protokolliert werden, welche Codezeilen durchlaufen wurden. Diese Testabdeckung kann mit lcov (http://ltp.sourceforge.net/coverage/lcov.php) aufbereitet und anschaulich dargestellt werden.

### CppCheck

CppCheck (http://cppcheck.sourceforge.net/ ) führt eine statische Codeanalyse durch und untersucht den Code auf verdächtige Konstrukte, die oft zu schwer entdeckbaren Fehlern im Betrieb führen.

### CCCC

Durch die Berechnung von Metriken gibt "CCCC - C and C++ Code Counter" (http://cccc.sourceforge.net/ ) Hinweise auf Codestellen, die nicht den üblichen Regeln für "ordentlichen Code" entsprechen und überprüft werden sollten.

## Das Software Engineering Praktikum auf dem Privatrechner

Die eigentliche Entwicklungsarbeit für das Praktikum werden Sie auf Ihrem Privatrechner durchführen. Dazu müssen Sie die benötigte Software selbst installieren und bei Problemen kann kein Support geboten werden. Mit den üblichen Foren und etwas Initiative lässt sich die benötigte Software aber installieren. Folgende Software wird benötigt:

- Git (https://git-scm.com)
- evtl. ein grafischer Git-Client (z.B. https://tortoisegit.org/)
- VPN-Software: Lesen Sie dazu die Anleitung von IT-Services vom Oktober 2020: https://its.h-da.io/infra-docs/docs/vpn.html Dort steht alles zur Installation. Wichtig: Der VPN-Endunkt hat seit Oktober 2020 die Adresse **vpn.fbi.h-da.de** Achtung! Eine VPN-Verbindung kann nur aus einem anderen Netz aufgebaut werden. Es funktioniert nicht innerhalb der Hochschule (Da braucht man es aber auch nicht)!
- Für Windows: Cygwin mit g++, make, gdb und cmake (https://www.cygwin.com/)
- CLion als C++ IDE von JetBrains (https://www.jetbrains.com/clion/
  Um die IDEs der Firma Jetbrains zu verwenden, benötigen Sie eine Lizenz. Diese erhalten Sie als Studierende kostenlos. Hierzu können Sie sich unter folgendem Link mit ihrer h-da Adresse registrieren: https://www.jetbrains.com/student/).
  Hinweis: Der Einsatz anderer IDEs ist prinzipiell möglich. Allerdings müssen Sie dann die Konfiguration des Projekts und der Testumgebung selbst vornehmen. Das sollten Sie nur tun, wenn Sie wirklich genau wissen, wie man das macht!
- GoogleTest https://github.com/google/googletest/releases
  Wenn Sie CLion als IDE verwenden, übernimmt CLion die Installation. Falls Sie eine andere IDE verwenden, müssen sie wahrscheinlich die entsprechende Bibliothek auf Ihren Rechner selbst kompilieren und in das Projekt einbinden. Verwenden Sie in diesem Fall unbedingt zum Erzeugen der Testbibliothek die gleiche IDE wie für Ihr eigentliches Projekt.

## Das Projekt "CocktailPro Legacy"

Im Jahr 2015 wurde eine Software zur Simulation eines Cocktail-Misch-Geräts entwickelt: der "CocktailPro". Die Software wurde aber nicht für einen dauerhaften Betrieb entwickelt. Deshalb enthält der Code praktisch keine Kommentare und leider auch keine sonstige Dokumentation. Auch kleinere Fehler sind bekannt, wurden aber nie korrigiert.

Nun soll die Software von Ihnen weiterentwickelt werden, allerdings so, dass die üblichen Qualitätsansprüche an eine professionelle Software erfüllt sind.

Sie erhalten zu Beginn des 2. Praktikums den vollständigen und kompilierbaren Code des CocktailPro.

Ihre Aufgabe im gesamten Praktikum besteht darin, das System wartbarer zu machen und um neue Funktionen zu erweitern.

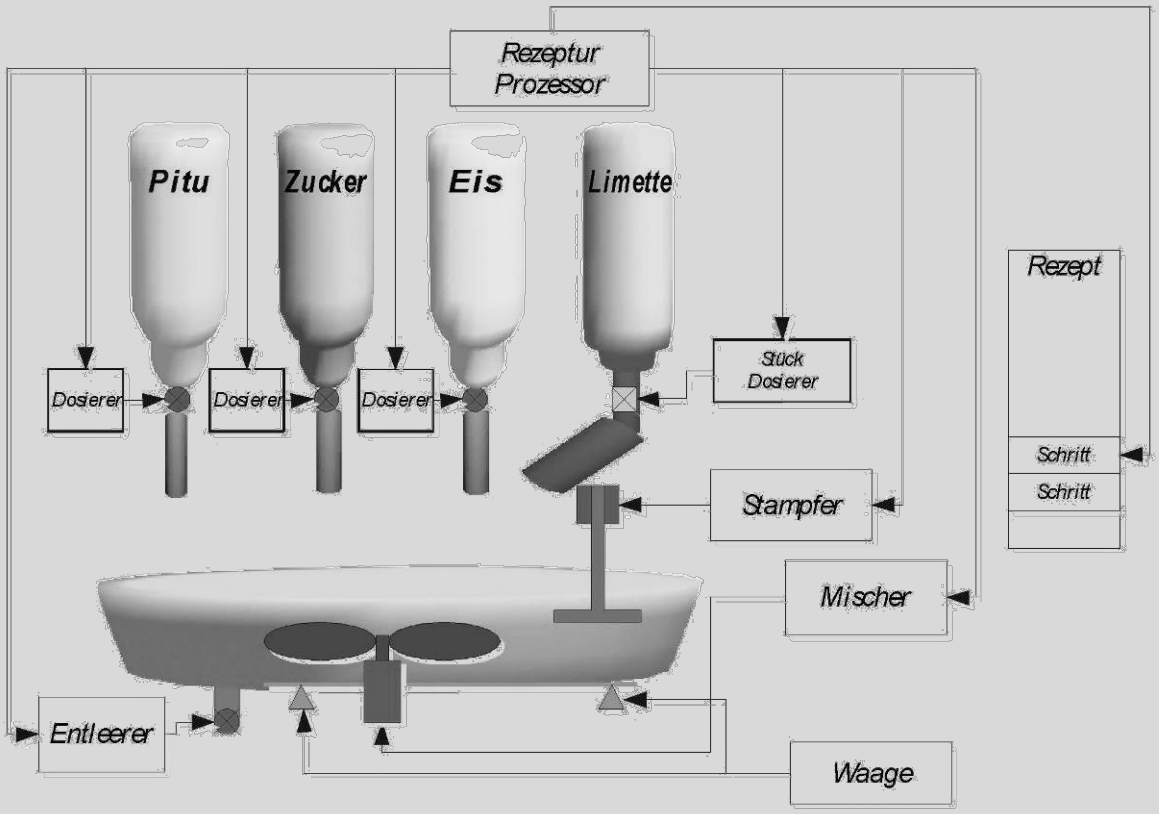
Ein Grundverständnis der Funktionsweise des CocktailPro's vermittelt die folgende Beschreibung.

Prof. Dr. Ralf Hahn
Prof. Dr. Urs Andelfinger
Prof. Dr. Stefan T. Ruehl
Michael Guist
Lukas Köhler

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Der Auftraggeber beschrieb 2015 aus seiner Sicht die Funktionsweise der Anlage:

## CocktailPro (2015)




**Entwickeln sie ein System für Restaurants zum automatisierten Zubereiten von Cocktails.**
Ein Kellner wählt ein Rezept aus. Anschließend werden die in den Rezept-Schritten angegebenen Zutaten (z.B. Pitu, Zucker, Eis und Limettenstücke für einen Caipirinha) in der richtigen Menge und Reihenfolge in einen Behälter gefüllt, gemäß Rezept für eine definierte Zeit gestampft, gerührt und anschließend entleert.
Der schematische Aufbau der technischen Anlage ist in der obigen Abbildung dargestellt.

Für jede Zutat ist eine Dosierstation vorhanden. Ein Dosierer liefert auf ein Kommando hin eine bestimmte Menge der Zutat. Ein Stückdosierer liefert eine abgezählte Anzahl von Stücken (z.B. Limetten).

Die dosierten Zutaten werden in einen Mischbehälter gefüllt. Im Mischbehälter kann die Mischung gerührt oder gestampft werden. Die Stampf- bzw. Rühreinrichtung kann für eine Zeitdauer aktiviert werden. Der Mischbehälter besitzt ein Entleerventil, mit dem der Cocktail in ein Glas entleert werden kann. Um den Dosiervorgang zu überwachen, steht der Mischbehälter auf einer Waage, die den Inhalt wiegt. Der Mischbehälter besitzt eine Vorrichtung zum Reinigen.
Ein Dosierer besitzt jeweils ein Ventil das geöffnet und geschlossen werden kann. Die Menge der bereits dosierten Zutat wird über eine Waage kontrolliert.
Durch die Befüllung der Dosierer mit Zutaten werden implizit die mischbaren Rezepte festgelegt. In der Regel sind einige Rezepte des Rezeptbuchs nicht mischbar, weil eine Zutat fehlt. Diese Rezepte werden erst gar nicht angeboten, sondern aussortiert.

**Ablauf "Mischen eines Cocktails" (Beispiel)**

Das System ist gestartet

1. Die (mit den vorhandenen Zutaten mischbaren) Cocktails werden als Auswahl angeboten.
2. Der Benutzer wählt einen Cocktail aus.
3. Entsprechend der Rezeptvorgaben werden Kommandos an Dosierer geschickt.
4. Das Ventil des Dosierers X wird geöffnet.
5. Wenn die Waage das gewünschte Zusatzgewicht (Delta) erreicht oder überschritten hat, wird das Ventil des Dosierers X geschlossen.
6. Analog zu Schritt 4., 5. und 6. werden die anderen Zutaten dosiert.
7. Der Mixer und/oder Stampfer wird für eine bestimmte Zeit aktiviert.
8. Der Cocktail wird entleert.
9. Die Reinigung des Mischbehälters erfolgt.
10. Goto 1.

## Bedienung

Das System soll hinter dem Tresen eines Restaurants installiert werden und wird nur vom Personal bedient. Im Prinzip erfolgt die Bedienung genau wie bei einer Espressomaschine in einem Restaurant. Das Konfigurieren der vorhandenen Zutaten soll der Wirt selbst erledigen können.

## CocktailPro-Simulator

Zu Testzwecken wird die fehlende Hardware durch Software-Simulationen ersetzt. Das heißt, anstatt eine echte Waage und Dosierer anzusteuern, werden diese mit Software simuliert. Statt eines Cocktails wird Text auf dem Bildschirm ausgegeben. Das eigentliche Mixen der Cocktails soll aber in der Simulation genauso ablaufen wie auf einem echten Zielsystem. Auf dem Zielsystem müssen später „nur noch“ die Simulations-Funktionen durch Funktionen mit gleichem Namen und Interface ersetzt werden, die dann allerdings auf die echte Hardware zugreifen.

**Vorgaben:**

- Dosierung
  - Werden Limettenstücke dosiert, fällt jede Sekunde ein Stück auf die Waage. Simulieren Sie so, dass jedes Stück 10g wiegt.
  - Eis wird nach Gewicht dosiert. Jede Sekunde wird ein Eiswürfel dosiert. Simulieren Sie, dies mit 20g pro Eiswürfel - Sie können nur Vielfache von 20g genau dosieren.
  - Bei Flüssigkeiten und feinkörnigen Zutaten (z.B. Zucker) ändert sich das Gewicht gleichmäßig. Simulieren Sie dies mit 1g/0,25s.
  - Ist das Entleerventil geöffnet, so verringert sich das Gewicht der Waage schnell. Simulieren Sie dies durch eine Reduzierung um 25g/s bis der Wert Null erreicht ist.
- Das Abwarten von Zeit wird durch ein Wait-Statement simuliert. Am Bildschirm wird pro Sekunde ein '*' ausgegeben. Zu Testzwecken ist es möglich, die Zeit zu „beschleunigen“ (dann wird eine Sekunde durch eine 10-tel Sekunde simuliert).
- Der Wirt soll konfigurieren können, welche Zutat in welchem Dosierer enthalten ist. Eine Änderung der Zuordnung von Dosierern zu Zutaten ist ohne Neuübersetzung des Programms möglich.
- Das System soll nach dem Start eine Auswahl-Liste der mischbaren Cocktails anzeigen
- Ausnahmen werden nicht realisiert (Dosierer hängt, ist leer etc.).

# 1. Praktikumstermin: Kickoff und Git-Versionsverwaltung (lokal)

Das gesamte Software Engineering Praktikum soll die qualitätsbewusste Entwicklung in einem Industrieprojekt demonstrieren.
Im ersten Termin werden grundlegende Dinge zur Arbeitsvorbereitung erledigt: Sie erhalten Informationen zum Ziel des Projektes, zum Aufbau des Praktikums und Sie werden in Projektteams mit je 4 Personen eingeteilt. Anschließend erfolgt eine erste Einführung in die Versionsverwaltung mit git. Als Nachbereitung setzen Sie Git und CLion auf Ihren Privatrechnern auf und üben das (lokale) Arbeiten mit git. (Im 2. Praktikum wird dann der zugelieferte Code unter Versionsverwaltung gestellt und Sie üben das Arbeiten im Team an einem gemeinsamen Code.)

## 1.1. Organisatorisches

Bitte beachten Sie die folgenden beiden organisatorischen Hinweise:

- Stellen Sie sicher, dass Sie Ihre Zugangsdaten für Ihren Hochschul-Account kennen.
- Da wir in diesem Semester (WS 21/22) das Praktikum coronabedingt remote durchführen, erarbeiten Sie die Praktikumsaufgaben auf Ihrem Privatrechner (und nicht im Labor). Daher sollten Sie immer Ihren Rechner während des Praktikums verfügbar haben.

## 1.2. Kickoff des Praktikums

### 1.2.1. Zielsetzung & Grundlagen CocktailPro

Lesen Sie bitte aufmerksam die Beschreibung zum CocktailPro (Graue Box weiter oben). Ihre Betreuer stellen Ihnen entlang der Beschreibung die Zielsetzung und die Aufgabenstellung des Praktikums vor.

Ihr Projektteam soll ein Altprojekt - den "CocktailPro" - übernehmen und um neue Features erweitern. Dieses Projekt wurde als Prototyp ohne nennenswerte Qualitätsauflagen erstellt. Die Entwickler haben daher nicht sehr auf Qualität geachtet. In den nächsten „Arbeitstagen“ sollen Sie dieses Projekt so anpassen dass eine Weiterentwicklung mit neuen Features möglich ist.

Ihr Vorgesetzter genehmigt Ihnen eine Einarbeitungszeit von 6 Arbeitstagen (1 Arbeitstag = 1 Praktikumstermin + Nachbereitung; Praktikum 1-6), bevor die Neuentwicklung startet. In dieser Zeit sollen Sie das Projekt so aufbereiten, dass es bestimmten Qualitätsregeln entspricht und Sie mit dem Inhalt vertraut sind.

Anschließend sollen Sie neue Features implementieren. Hierfür stehen Ihnen weitere 4 Arbeitstage zur Verfügung (1 Arbeitstag = 1 Praktikumstermin + Nachbereitung: Praktikum 7-10),

Als zentrales Tool wird während der Entwicklung Git eingesetzt, als IDE CLion. Jenkins wird der zentrale Integrationsserver sein.

### 1.2.2. Teambildung für das Praktikum

Das Praktikum wird in Teams (i.d.R. 4er-Teams) durchgeführt. Die Gruppenbildung erfolgt jetzt gemeinsam mit Ihnen. Ihre Betreuer geben Ihnen weitere Anweisungen (z.B.bezüglich Breakoutrooms), wie die Gruppenbildung erfolgt. Es ist wichtig, dass Sie für das gesamte Semester in Ihrem Team bleiben, da wir die Gruppenzugehörigkeit fest in unsere Buildumgebung (git und Jenkins) eintragen werden.

Sollte es im Team zu unüberwindbaren Problemen bei der Zusammenarbeit kommen, informieren Sie bitte möglichst frühzeitig Ihre Betreuer. (Das kommt selten vor, aber es **kann** vorkommen. Bitte nehmen Sie in diesem Fall von sich aus Kontakt zu uns auf).

### 1.2.3 Git als Basis für die Arbeit im Team

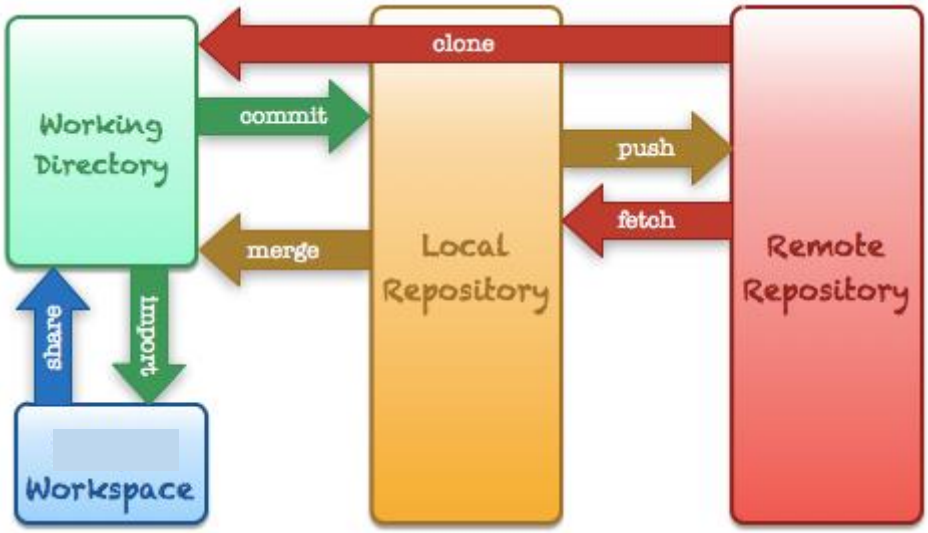
Vielleicht haben Sie schon in anderen Projekten bemerkt, dass es in einem Team ohne genaue Absprachen ein hohes Risiko gibt, dass mehrere Personen die gleiche Datei bearbeiten und dass dabei Arbeit verloren gehen kann. Dieses und andere Risiken löst man in Software-Projekten mit "Konfigurationsmanagement". Wir verwenden dafür das Tool "Git".

***Versionsverwaltung mit Git***

*Git ist eine Software zur Versionsverwaltung, die sich in der Softwareentwicklung mittlerweile am stärksten durchgesetzt hat (siehe z.B. github oder die Verwaltung der Linux-Quellen).*

*Git verfolgt das Konzept, dass es (vernetzte) **Repositories** (also Datenspeicher) gibt, die jeweils **alle** Versionen von Dateien und Verzeichnissen enthalten. Das **lokale** Repository enthält demnach auch alle Versionen.*

Ihre Betreuer erläutern Ihnen zunächst die prinzipielle Funktionsweise von Git (siehe Schaubild). Die Nachbereitungsaufgabe besteht darin, Git zu installieren und sich bis zum nächsten Praktikum mit der Nutzung von Git (lokal) vertraut zu machen.



## 1.3. Nachbereitung: Lokale Arbeitsumgebung

Die Nachbereitung umfasst folgende Aufgaben:

- Installation der Entwicklungstools
- Installation von CLion
- Installation des VPN
- Installation von git
- Übungsaufgaben zum lokalen Arbeiten mit git

### 1.3.1. Installation der Entwicklungstools

Hinweis: Falls Sie bereits Git und / oder CLion (C++) im Einsatz haben, prüfen Sie trotzdem die nachfolgenden Hinweise, damit Sie wirklich eine mit unseren Vorgaben kompatible Installation auf Ihren privaten Rechnern haben:

Installieren Sie die nachfolgenden Entwicklungstools! Stellen Sie sicher, dass folgende Tools auf Ihrem Rechner vorhanden sind[3]:

- g++ (Der Gnu-Compiler)
- gdb (Der Debugger zum Gnu-Compiler)
- make (Ein klassisches Buildtool)
- cmake (Ein fortgeschrittenes Buildtool)
- optional: doxygen und graphviz (Tool zur Erzeugung einer Dokumentation)

Unter Windows wird die Verwendung von Cygwin (https://www.cygwin.com/) empfohlen. Wählen Sie im Installationsprogramm g++, make, gdb und cmake - am besten installieren Sie auch noch doxygen und graphviz.
**Achtung**: Das Cygwin-Installationsprogramm ist manchmal etwas schwer zu verstehen. Sie müssen zuerst im Auswahlfenster den Filter auf "Full" setzen, damit Sie alle installierbaren Programme sehen. Dann suchen Sie die gewünschten Programme und wählen für jedes Programme die gewünschte Version (Sie können die Checkbox nicht anklicken!). Wählen Sie jeweils die neueste offizielle Version (keine Testversionen).

### 1.3.2. Installation CLion

Installieren Sie CLion auf dem Rechner auf dem Sie die Praktikumsaufgaben lösen werden. Besorgen Sie sich die (kostenlose) Lizenz für Studierende und registrieren Sie CLion.

### 1.3.3. Installation VPN

Installieren Sie den VPN-Zugang[4] auf dem Rechner auf dem Sie die Praktikumsaufgaben lösen werden. Lesen Sie dazu die Anleitung unter https://its.h-da.io/infra-docs/docs/vpn.html .

[3] Wenn Sie die Befehle in einer Konsole (z.B. in der Git Bash) eingeben, sollte das jeweilige Programm gefunden werden. Falls nicht: installieren Sie die Tools am besten mithilfe von Cygwin.
[4] Sie brauchen den VPN-Zugang ab dem 2. Praktikumstermin.

#### 1.3.4. Installation von Git:

Falls Sie Git noch installieren müssen: hier erhalten Sie die aktuelle Version von Git: https://git-scm.com.

##### Initiale Einrichtung Ihrer Git-Installation

Damit Änderungen besser nachvollziehbar werden, müssen Sie Ihre Git-Installation personalisieren. **Diese Personalisierung müssen Sie auf jedem Rechner (bzw. Account) durchführen, auf dem Sie an dem Projekt arbeiten.**

- Konfigurieren Sie Ihren Nutzernamen[5] und ihren Email-Account:
  `git config --global user.name "Ihr Name"`
  `git config --global user.email "emailadresse"`
  (Ersetzen Sie "**Ihr Name**" durch ihren Namen, und "**emailadresse**" durch ihre Email-Adresse.)
- Konfigurieren Sie einen Editor für Git:
  - Unter Linux: `git config --global core.editor "kate"`
  - Unter Windows: `git config --global core.editor "notepad"`

## 1.4. Nachbereitung: Lokales Arbeiten mit Git:

Arbeiten Sie die nachfolgenden Übungsaufgaben gründlich durch und beantworten Sie die enthaltenen Fragen. **Bringen Sie Ihr Vorbereitungsergebnis und ggf. Ihre Fragen zum nächsten Praktikumstermin mit. Sie müssen nichts hochladen, aber wir erwarten, dass Sie die Aufgaben wirklich durchgeführt haben. (Falls nicht, werden Sie in den kommenden Wochen große Probleme haben, das Praktikum erfolgreich zu absolvieren!).**

Die folgenden Aufgaben und Anweisungen beziehen sich auf eine Linux-Installation. Falls auf Ihrem Privatrechner Windows läuft: Auch damit können Sie die Aufgaben genau wie beschrieben durchführen: Starten Sie Git-Bash. Dann können Sie die Befehle genauso in der "Git-Bash" eingeben, die zur Verfügung steht, sobald Sie Git installiert haben.

##### Erzeugen Sie ein lokales Git-Repository als Spielwiese

Zunächst lernen Sie Git als lokales Versionsverwaltungssystem kennen. Das bedeutet, dass alle Dateien und Versionen zwar in einem Repository verwaltet werden, aber dieses Repository liegt auf der lokalen Festplatte. (Später binden wir ein Repository auf einem zentralen Git-Server ein).

1. Öffnen Sie ein Terminal-Fenster („Konsole“) bzw. die Git-Bash
2. Erstellen Sie ein Verzeichnis mit dem Namen „dev“ (für „Development“).
   `mkdir dev`
   (Das Linux-Kommando mkdir steht für „make directory“)
3. Wechseln Sie in das Verzeichnis dev.
   `cd dev`
   (cd steht für "change directory"[6])
4. Erstellen Sie ein neues Verzeichnis, dessen Inhalt durch git verwaltet wird:
   `git init project1`

[5] Verwenden Sie bitte "Nachname, Vorname" für ALLE Ihre Git-Installationen (z.B. auf dem Laborrechner und auf Ihrem Privatrechner).
[6] Wenn Sie in das übergeordnete Verzeichnis wechseln wollen, dann geben Sie `cd ..` ein

5. Wechseln Sie in das Verzeichnis
   `cd project1`
6. Lassen Sie sich den Inhalt des Verzeichnisses anzeigen:
   `ls`
   Sie sehen, dass im Verzeichnis noch nichts enthalten ist.
   Wenn Sie „`ls -al`“ eingeben, zeigt Linux auch versteckte Dateien und Verzeichnisse an (die mit einem "." beginnen).
   Dann werden Sie sehen, dass es ein Verzeichnis mit dem Namen „.git“ gibt; hier ist das Git-Repository "versteckt".

### Stellen Sie eine neue Datei unter (lokale) Versionsverwaltung durch Git

Als nächstes erstellen Sie eine Textdatei und fügen sie zur Versionsverwaltung hinzu.

1. Öffnen Sie einen Editor und schreiben Sie einen kurzen Text.
2. Speichern Sie die Datei mit dem Namen „datei1.txt“ im „project1“ Verzeichnis
3. Führen Sie das Kommando „`git status`“ auf der Kommandozeile im Verzeichnis „project1“ aus. Interpretieren Sie die Ausgabe. Was bedeutet die Ausgabe?

Die Datei steht nicht unter Versionskontrolle

4. Führen Sie das Kommando aus:
   `git add datei1.txt`
   Führen Sie erneut „`git status`“ aus und interpretieren Sie das veränderte Ergebnis:

Die Datei wird zum VC hinzugefügt

5. Führen Sie den Befehl aus:
   `git commit -m "Ein Kommentar"`
   Was zeigt der Befehl „`git status`“ jetzt an?

Es wird gezeigt, dass die Datei gestaged wurde

6. Was zeigt der Befehl „`git log`“ an?

Die commit history des lokalen git

### Änderungen rückgängig machen

In dieser Aufgabe lernen Sie, wie lokale Änderungen mithilfe von Git rückgängig gemacht werden.

1. Verändern Sie die Datei „datei1.txt“ aus der vorherigen Aufgabe mit einem Editor
2. Speichern Sie die veränderte Datei ab
3. Lassen Sie sich mit „git status“ den Status anzeigen. (Die Ausgabe sollte besagen, dass „datei1.txt“ modifiziert wurde)
4. Stellen Sie mit Git die Datei "datei1.txt" wieder so her, wie der Zustand nach dem Commit aus der vorherigen Aufgabe war. Finden Sie das Kommando selbst heraus und schreiben Sie es auf:

git checkout [Dateiname]

### Weiterführende Hinweise zu Git (WICHTIG!)

Wenn Sie die grundlegenden Git-Kommandos verstanden haben, dann können Sie zukünftig auch die Git-Befehle innerhalb der IDE oder andere Plugins verwenden.

Es gibt noch viele weitere Befehle und Möglichkeiten zur Nutzung von Git. Bei Problemen bietet die Rückmeldung von Git erste wichtige Hinweise. Oft steht in der Meldung sogar genau der Befehl, den Sie zur Lösung des Problems brauchen. Man muss es nur lesen.

Hier noch einige nützliche Links zum Thema Git:

https://git-scm.com/book/en/v2/Getting-Started-Git-Basics

https://rogerdudler.github.io/git-guide/index.de.html

https://www.atlassian.com/git/tutorials/setting-up-a-repository

https://marklodato.github.io/visual-git-guide/index-en.html

https://learngitbranching.js.org/