6. Praktikumstermin: Meilenstein und Reflektion

Der Zwischenstand, der die Basis für die Weiterentwicklung der Anwendung in den restlichen Terminen sein wird, ist nach diesem Termin und seiner Nachbereitung erreicht. Deshalb nehmen wir uns etwas Zeit, um das erreichte Ergebnis zu bewerten.

6.1. Vorbereitung

- Prüfen Sie die Ergebnisse von Doxygen, CppCheck, CCCC und GoogleTest.
- Es gibt einen automatisierten Test für jede (testbare) Methode (außer für die Klassen CocktailPro und CocktailZubereiter. Die Tests laufen in wenigen Sekunden (mit Turbo).
- Ein lauffähiger und testbarer Entwicklungsstand ist in Git eingecheckt. Prüfen Sie lokal, ob Ihr Cocktailautomat wirklich noch so funktioniert, wie es Ihr Auftraggeber wünscht.

6.2. Aufgaben

6.2.1. Bestandsaufnahme der Qualität

Analysieren Sie im Team die Jenkins-Ausgaben zu den bisher durchgeführten Maßnahmen. Überlegen Sie, welche Aufgaben noch zu erledigen sind.

Betrachten Sie die einzelnen Arbeitsergebnisse und überlegen Sie, welche Qualitätseigenschaften diese Maßnahmen verbessert haben.

6.2.2. Code-Coverage / Test-Coverage

Es gibt die Möglichkeit während der Ausführung von Code zu protokollieren, welche Codezeilen bzw. Anweisungen während der Ausführung der Tests ausgeführt wurden. So entsteht eine "Testabdeckung (Coverage)". Diese wird bereits während der automatischen Testdurchführung erfasst und von Jenkins (und dem Tool lcov) aufbereitet.
Vielleicht haben Sie in Ihrer Jenkins-Oberfläche schon den Eintrag "Code Coverage" entdeckt?

- Analysieren Sie die Ausgabe der Testabdeckung und klicken Sie links unter "Directory" auf das Projektverzeichnis.
- Die Einträge in den Spalten sagen jeweils aus, wie viele der Klassen / Zeilen / Funktionen während der automatischen Tests durchlaufen wurden - und wie viele es insgesamt gibt.

Filename	Line Coverage		Functions	
Recipe.cpp	50.0 %	15 / 30	60.0 %	6 / 10
RecipeBook.cpp	85.3 %	93 / 109	100.0 %	8 / 8
RecipeStep.cpp	62.5 %	10 / 16	50.0 %	3 / 6

  Von der Datei RecipeBook.cpp wurden hier also 93 von 109 Zeilen durchlaufen - und 8 von 8 Funktionen erreicht. Die Klasse wurde also schon weitgehend getestet.
  Klicken Sie auf "RecipeBook.cpp" und analysieren Sie die Darstellung und suchen Sie die getesteten Zeilen[12].
- Finden Sie heraus, warum einige Methoden bereits Überdeckungswerte anzeigen, obwohl es gar keine Tests dafür gibt.

6.2.3. Weitere Qualitätsverbesserung

[12] Die Zählung der Aufrufe ist anfangs oft nicht ganz korrekt. Das ist aber nicht weiter schlimm, weil das Problem verschwindet, wenn man die Fehlerüberdeckung erhöht.

Gehen Sie die Punkte an, die Sie in der Bestandsaufnahme zu Beginn der Übung entdeckt haben.

- Verbessern Sie weiter Ihre Testabdeckung - nun dürfen Sie auch die Klassen CocktailPro und CocktailZubereiter testen.

6.3. Nachbereitung

Teilen Sie die verbleibenden Tätigkeiten innerhalb Ihres Teams auf. Folgende Abnahmepunkte müssen bis zum Abgabetermin für den nächsten Praktikumstermin erfüllt sein:

- Testen Sie jetzt auch die Klassen CocktailPro und CocktailZubereiter gemäß der bekannten Regeln.
- Betrachten Sie Ihre Klassen und Methoden mit niedriger Coverage und überlegen Sie, welche Tests fehlen. Ergänzen Sie diese Tests, um die Testabdeckung in Richtung 100% zu bringen. Die "Line Coverage" muss mindestens 90% betragen. Vielleicht schaffen Sie sogar 100%!?
- Ein testbarer Entwicklungsstand mit deutlich erhöhten Testabdeckung ist in Git eingecheckt.
- Ein lauffähiger Entwicklungsstand ist in Git eingecheckt und erfüllt alle bisherigen Vorgaben. Sie können dies mit Jenkins leicht überprüfen.