Prof. Dr. Ralf Hahn
Prof. Dr. Jens-Peter Akelbein
Michael Guist

Praktikum SWE

WS2015

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Version 3.2

Praktikumsaufgabe Software Engineering

Ablauf des Praktikums

Das Praktikum besteht aus 5 Terminen. In diesen Terminen wird die Software zu einer **Cocktailmischanlage „CocktailPro"** von der Anforderungsanalyse bis zum fertigen Programm entwickelt (Natürlich stark vereinfacht, ohne Fehler- und Ausnahmebehandlung, ohne GUI, ohne Datenbankanbindung, ohne Anbindung an eine reale Cocktailmischanlage, ohne Ressourcenbeschränkung).
Das Praktikum setzt voraus, dass Sie mit den Grundlagen der Modellierung und der Bedienung des Case-Tools MagicDraw vertraut sind. Ist dies nicht der Fall, sollten Sie die Übung aus OOAD zur Auffrischung noch einmal durcharbeiten und sich die Anleitung für MagicDraw unter https://www.fbi.h-da.de/labore/case/laborausstattung/magicdraw/einfuehrung-magicdraw.html durchlesen.

In dem Praktikum besteht Anwesenheitspflicht. Unentschuldigtes Fernbleiben führt zum Ausschluss und Nichtbestehen des Praktikums. Die im jeweiligen Praktikum durchzuführenden Tätigkeiten sind vor dem Praktikumstermin vorzubereiten.
Falls eine Gruppe unvorbereitet zum Praktikum erscheint oder zu Beginn des darauf folgenden Praktikums nicht die zum vorherigen Termin geforderte Leistung erbracht hat, wird einmalig ermahnt (GELBE KARTE). Beim 2. Vorkommen dieser Art wird der Studierende vom weiteren Besuch der Veranstaltung ausgeschlossen und das Praktikum als nicht bestanden gewertet (ROTE KARTE). Eine Zulassung zum Leistungsnachweis ist dann nicht möglich, das Praktikum muss in einem folgenden Semester wiederholt werden.

Warnungen

- In diesem Praktikum geht es um Software Engineering. Deshalb stehen das Vorgehen, Modelle, Entwurfsmuster etc. im Vordergrund der Aufgaben. Dass am Schluss ein lauffähiges Programm entsteht, das die Anforderungen erfüllt, wird als selbstverständlich angesehen. Für die Endabnahme müssen Sie demonstrieren, dass Sie diese Techniken und Ihren Code beherrschen.
- In diesem Praktikum arbeiten Sie weitgehend selbständig und werden nur wenig überwacht. Insbesondere ist die Abnahme für die ersten Termine recht freizügig. Sie könnten die Abnahme für die Anforderungsanalyse prinzipiell auch erhalten, wenn Sie einen einzigen Use Case modellieren, der einen Cocktail mischt. Bei der Klassenmodellierung werden Sie dann allerdings feststellen, dass Sie die Anforderungen nicht hinreichend genau kennen und die Arbeit dort nachholen. Das Gleiche passiert, wenn Sie Sequenzdiagramme vernachlässigen und

sofort mit der Entwicklung starten. Das Vernachlässigen der Ergebnisse einer Phase führt erfahrungsgemäß zu deutlich höherem Aufwand in den folgenden Phasen ("10-er Regel", weil sich der Aufwand verzehnfacht). Das ist zwar sehr ineffizient aber dafür umso lehrreicher...

- Die Implementierung erfordert die Beherrschung von C++. Sie sollten jedenfalls mit Klassen, Vererbung, Konstruktoren und Pointern vertraut sein. Ansonsten ist das Praktikum kaum zu bestehen.

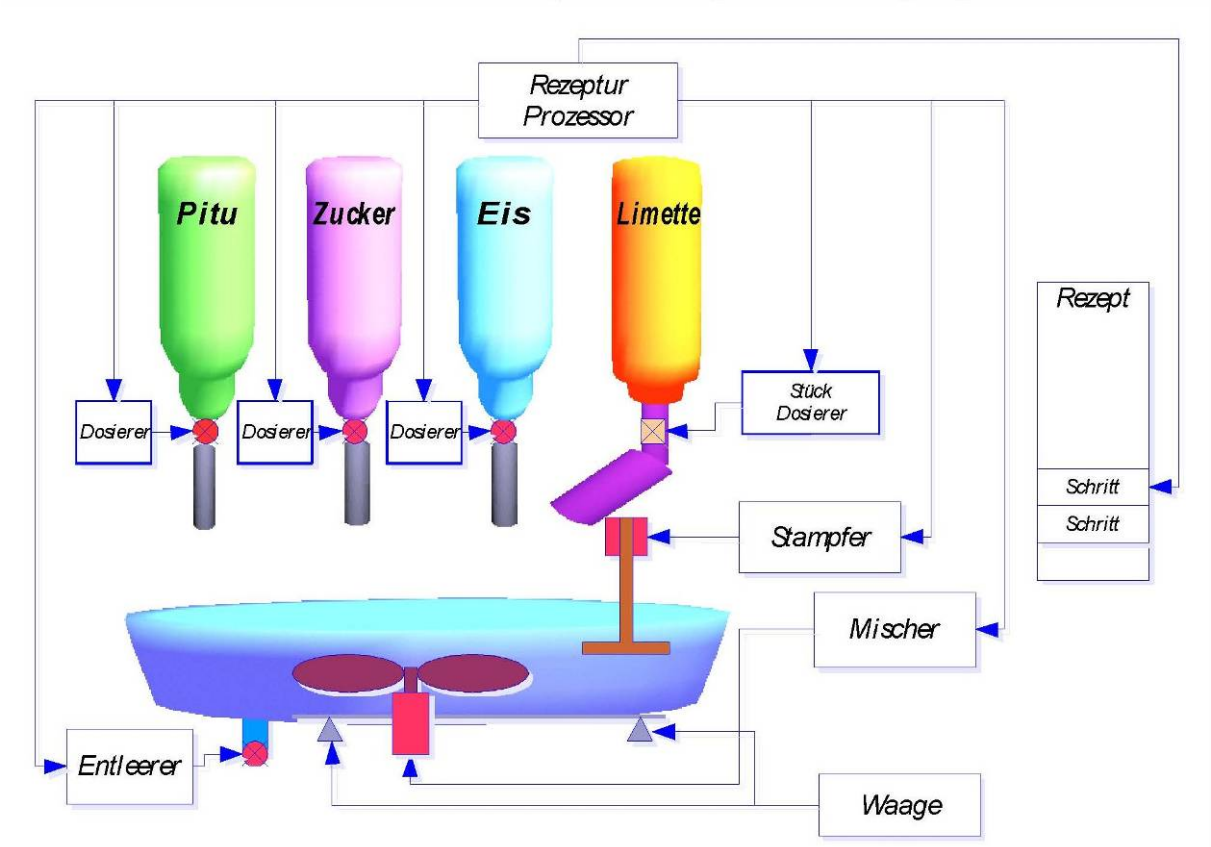
Nutzung von MagicDraw

Die Bedienung von MagicDraw wurde in OOAD bereits geübt. Zur Klärung von Fragen zur Bedienung ist es sinnvoll, die Praktikumsaufgaben des vorigen Semesters und die Anleitung für MagicDraw vorliegen zu haben: https://www.fbi.h-da.de/fileadmin/Labor/case/MagicDraw/MagicDraw-Anleitung.pdf

Lastenheft CocktailPro

Der Auftraggeber beschreibt aus seiner Sicht die Funktionsweise der Anlage:

Entwickeln sie ein System zum automatisierten Zubereiten von Cocktails. Ein Benutzer 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 Vorgabe im Rezept für eine definierte Zeit gestampft, gerührt und anschließend entleert.
Der schematische Aufbau der technischen Anlage ist in der folgenden 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 entleert werden

kann. Um den Dosiervorgang zu überwachen, steht der Mischbehälter auf einer Waage, die den Inhalt wiegt. Der Mischbehälter besitzt eine Vorrichtung zum automatischen Reinigen.

Die Dosierstation

- Eine Dosierstation besitzt jeweils ein Ventil das geöffnet und geschlossen werden kann. Die Menge der bereits dosierten Zutat muss über die Waage kontrolliert werden.
- Der Stückdosierer funktioniert genauso wie die „normalen" Dosierer.
- Durch die Befüllung der Dosierstationen mit Zutaten werden die mischbaren Rezepte festgelegt (in der Regel sind nicht alle Rezepte des Rezeptbuchs mischbar).

Der Rezepturprozessor

- Die Verarbeitungsschritte sind in einem Rezept (s. u.) festgelegt.
- Der Rezepturprozessor führt, falls notwendig, koordinierende Tätigkeiten aus.

Das Rezept

- Das Rezept ist eine geordnete Folge von Rezeptschritten, die beim Herstellen des Getränks durchgeführt werden müssen.

Die Rezeptschritte

- Die Rezeptschritte sind die einzelnen Schritte des Rezepts.
- Jeder Rezeptschritt enthält die Information welche Station den Schritt ausführen soll und den Parameter, den sie dazu benötigt (Menge / Anzahl / Zeitdauer).

Das Rezeptbuch (EXTERNE ZULIEFERUNG!!!)

- Das Rezeptbuch entspricht einem Buch der Art "Die 1000 Cocktails der Welt" und verwaltet eine Menge von Rezepten und den dafür benötigten Zutaten und Dosierungen
- Es liefert eine nummerierte Liste mit Rezepten
- Zu einem (über die Nummer) ausgewählten Rezept kann der i-te Rezeptschritt abgefragt werden. Es wird die erforderliche Zutat (bzw. der Verarbeitungsschritt) und die Menge bzw. Verarbeitungsdauer geliefert.
- Für die Entwicklung erhalten Sie eine vorläufige Version des Rezeptbuchs, das nur ein paar Rezepte enthält. Später soll ein echtes Rezeptbuch mit ca. 1000 Cocktails zugeliefert werden.

Die Waage

- Die Waage liefert 2 Gewichtswerte. Das absolut gewogene Gewicht und ein Differenzgewicht „Delta", das relativ zu einem frei setzbaren Delta-Nullpunkt ist.
- Die Waage ist so geeicht, dass das Absolutgewicht mit dem leeren Mischbehälter 0 ist.
- Der „Delta-Nullpunkt" kann durch einen Aufruf auf das aktuelle Absolutgewicht gesetzt werden

Der Mischer

- Der Mischer mischt für eine definierte Zeitdauer.

Der Stampfer

- Der Stampfer stampft für eine definierte Zeitdauer

Der Entleerer des Mischbehälters

- Der Mischbehälter verfügt über ein Ventil zur Entleerung („Entleerer") das geöffnet und geschlossen werden kann. Die Restmenge im Mischbehälter muss über eine Waage kontrolliert werden.
- Der Mischbehälter reinigt sich automatisch nach der vollständigen Entleerung

Das Display

- Das Display bietet die Cocktails zur Auswahl und gibt Informationen zum Mischvorgang aus.

Szenario

Mischen eines Cocktails

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

Bedienung

Das System soll hinter dem Tresen installiert werden und wird nur vom Personal bedient. Im Prinzip erfolgt die Bedienung genau wie bei einer Espressomaschine in einem Restaurant.

Randbedingungen

- Das Rezeptbuch wurde bereits entwickelt und liegt als Binärkomponente mit C++ Header vor. Außerdem gehört dazu eine Beispiel-Anwendung (main.cpp) und eine Dokumentation des zugehörigen MagicDraw-Modells.
- Es wird Quellcode zum Einlesen einer Datei bereitgestellt.
- Es wird Quellcode zum Warten von x msec. bereitgestellt.
- Es wird Quellcode bereit gestellt, der die Verwendung der Standardbibliothek <list> zeigt
- Es wird eine Vorgängerversion als Demo bereitgestellt. Sie hat zwar nicht vollständig die gewünschte Funktionalität, aber sie veranschaulicht das Verhalten des Systems.

Aufgabenstellung

In der vorhergehenden Beschreibung der zu realisierenden Anlage hat der Kunde aus seiner Sicht und mit seinen Beschreibungsmitteln die Funktionsweise der Anlage beschrieben.

Wir wollen nun diese Beschreibung umsetzen in eine präzise mit UML beschriebene Spezifikation, die dann anschließend mit dem Kunden durchgesprochen wird (bei uns simuliert durch ein Anforderungs-Review, das durch eine anderer Praktikumsgruppe durchgeführt wird).
Im Rahmen des Praktikums wird ein Prototyp des Systems entwickelt. Dieser verwendet Software-Simulationen der Geräte wie Dosierer etc. und läuft komplett auf einem Standard-Rechner.

Der Prototyp soll so realitätsnahe wie möglich entwickelt werden. Dazu wird die fehlende Hardware durch Software-Simulationen ersetzt, die Sie u. a. selbst entwickeln sollen. (Das heißt, anstatt eine echte Waage anzusteuern, simulieren Sie diese mit Software.
Die Simulations-Funktionen simulieren die fehlende Hardware und liefern die Informationen, wie sie von der echten Hardware erwartet würden oder die Simulations-Funktionen nehmen die Informationen an und geben sie am Bildschirm aus, statt sie an die Hardware (z. B. Anzeigegeräte) weiterzugeben. Durch die Verwendung dieser Simulations-Funktionen kann das eigentliche Mixen der Cocktails in der Simulation genauso ablaufen wie auf einem echten Zielsystem. Auf dem Zielsystem müssten später „nur noch" die Simulations-Funktionen durch Funktionen mit gleichem Namen und Interface ersetzt werden - die dann allerdings auf die echte Hardware zugreifen.

Vorgaben:

1. Der CocktailPro enthält 10 Dosierer + 1 Mixer + 1 Stampfer
2. Die im Prototyp nicht vorhandene Hardware (Dosierer, Waage usw.) wird durch entsprechende Funktionen simuliert,
3. Ausgaben von simulierten physischen Geräten (z.B. Waage) sollen über Bildschirmausgaben (cout) sichtbar gemacht werden.
4. Sämtliche Aktionen von Geräten werden durch entsprechende Bildschirmausgaben protokolliert (z.B. „Ventil von Dosierer O-Saft geöffnet").
5. Auf dem Display der Waage wird sowohl das absolute als auch das Delta-Gewicht angezeigt
6. Solange ein Ventil geöffnet ist, soll sich das Gewicht auf der Waage selbstständig erhöhen.
   - 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.

7. Das Abwarten von Zeit wird durch ein Wait-Statement simuliert. Am Bildschirm wird pro Sekunde ein '*' ausgegeben. Zu Testzwecken ist es möglich, die Wartezeit zu „kürzen" (dann wird eine Sekunde durch eine 10-tel Sekunde simuliert).
8. Der Wirt soll konfigurieren können, welche Zutat in welchem Dosierer enthalten ist. Während des Systemstarts wird eine Text-Datei „zutaten.txt" über ein Programm ähnlich dem zugelieferten Programm „readfile.cpp" eingelesen, welche die Zuordnung von Dosierern zu Zutaten festlegt (Datei wird bereitgestellt). Dadurch ist eine Änderung der Zuordnung von Dosierern zu Zutaten ohne Neuübersetzung des Programms möglich.
9. Das System soll nach dem Start eine Auswahl-Liste der mischbaren Cocktails anzeigen (Im Prototypen soll auch die Konfiguration (Dosierer: Zutat) ausgegeben werden). Es werden nur die Cocktails zur Auswahl angeboten, für die auch alle Zutaten vorhanden sind.
10. Ausnahmen werden nicht realisiert (Dosierer hängt, ist leer etc.).
11. Um zukünftige Erweiterungen zu erleichtern, soll die Kommunikation zwischen der Waage und den Beobachtern der Waage (Dosierer, Mischbehälter und dem Display der Waage) nach dem Beobachter-Muster erfolgen.

Hinweise

- Die vorhergehende Beschreibung des Systems gibt eine mögliche Aufteilung in Klassen (aus Sicht des Auftraggebers) vor. Prüfen Sie eingehend, ob diese Klassen wirklich sinnvoll sind.
- Analysieren Sie die Rezepte und deren Inhalt genau. Gibt es logische Unterschiede bei den Zutaten, die Sie auch entsprechend modellieren sollten?
- Eine der kniffligsten Stellen im gesamten System ist die Interaktion zwischen den Dosierern / Display und der Waage. Da das Gewicht nicht tatsächlich erhöht wird, muss die Gewichtserhöhung simuliert werden. Überlegen Sie genau, wo und wie Sie diese Funktionalität umsetzen.
- Überlegen Sie sich wie der Systemstart ablaufen soll und wie die Konfiguration eingelesen wird.
- Denken Sie an die Entwurfs-Grundprinzipien (Geheimnisprinzip, starke Kohäsion, schwache Kopplung), um einen möglichst stabilen Entwurf zu erhalten. Denken Sie auch an naheliegende Änderungen im System (insbesondere Änderungen der Rezepte und Zutaten).
- Schauen Sie sich die Schnittstellen des zugelieferten Rezeptbuchs an, bevor Sie mit Ihrem Entwurf beginnen.
- Es wird Source-Code („readfile.cpp") zum Einlesen der Datei „zutaten.txt" bereitgestellt, den Sie verstehen und für Ihren Bedarf modifizieren sollen.
- Es ist oft sinnvoll Standard-Datencontainer wie z.B. LIST zu verwenden. Ein Code-Beispiel für die Verwendung von LIST wird bereitgestellt.

- In der Endabnahme wird eine veränderte Zutatendatei eingespielt. Ihr System muss diese einlesen, dann die mischbaren Cocktails im Auswahlmenü anzeigen und 1-2 Cocktails richtig mischen. Dabei wird auf die Einhaltung der vorgegebenen Stückelung (vgl. Vorgabe 6 auf Seite 7) und Geschwindigkeit geachtet.
- In der Endabnahme wird ein Cocktail im normalen langsamen Mischmodus gemischt und ein Cocktail im beschleunigten Modus (10-fache Geschwindigkeit vgl. Vorgabe 7 auf Seite 8). Es wird Source-Code („Timer.cpp") zum Abwarten einer definierten Zeitspanne bereitgestellt, den Sie verstehen und für Ihren Bedarf modifizieren sollen.
- Es gibt für dieses Praktikum viele richtige Lösungen. Ihre Aufgabe ist es – so wie in einem echten Projekt auch – einen geordneten und sicheren Weg zu Ihrer Lösung zu finden. Sie sollen während des Praktikums ein Gefühl dafür bekommen, wann eine Entwicklungsphase hinreichend gründlich bearbeitet wurde.

Hinweise zur Technik

- Textdateien (z.B. .csv oder .txt) unter Windows und Linux haben ein unterschiedliches Dateiformat und werden von manchen Editoren beim Speichern automatisch in das entsprechende Format des Betriebssystems gewandelt. Wenn Sie also die Datei "zutaten.txt" unter Windows anschauen und speichern und anschließend die Datei mit Ihrem Code nach Linux kopieren, liegt dort das Windows-Format vor und es kommt beim Einlesen der Texte zu seltsamen Effekten (Strings sind leer etc.). Die Unixbefehle dos2unix bzw. unix2dos wandeln die Dateiformate.Diverse Editoren wie z.B. Notepad++ bieten außerdem die Möglichkeit zur Auswahl eines Speicherformats.
- Wenn Sie eine Folge von Sternen o.ä. in einer Zeile ausgeben wollen, müssen Sie bei ANSI-C++ den Ausgabepuffer nach jeder Ausgabe mit cout << .... << flush explizit leeren.

Aufgaben

1. Termin: Anforderungsanalyse

Vorbereitung

- Arbeiten Sie die vorangehende Systembeschreibung und die Hinweise genau durch. Dokumentieren Sie aufkommende Fragen.
- Erstellen Sie den Use Case "Cocktail mischen" mit einer textuellen Beschreibung auf einem Blatt Papier. Dazu frischen Sie Ihr Wissen auf[1], über die textuelle Beschreibung von Use Cases und UML Use Case Diagrammen.
- Hinweis: Das Praktikum setzt voraus, dass Sie mit der Bedienung des Case-Tools MagicDraw vertraut sind. Ist dies nicht (mehr) der Fall, sollten Sie die Übung aus OOAD zur Auffrischung noch einmal durcharbeiten und in der Anleitung zu MagicDraw den Abschnitt zu den Use Case-Diagrammen lesen.

1.1.) Anwendungsfälle (Use-Cases)

Analysieren Sie die vorhergehende Systembeschreibung auf Anwendungsfälle, die das System erfüllen muss. Erstellen Sie ein Use-Case-Diagramm und beschreiben Sie die einzelnen Use-Cases im Modell.

1.2.) Einbinden des Rezeptbuchs

Das Rezeptbuch wird von Extern zugeliefert[2].
Der Betreuer erklärt Ihnen, wie Sie die Zulieferung in Ihr Modell integrieren.

1.3.) Anforderungs-Review

Der Betreuer zeigt Ihnen, wie Sie ein Dokument mit Ihren Arbeitsergebnissen erzeugen. Prüfen Sie den Inhalt und drucken Sie dann das Dokument aus.
Setzen Sie sich mit einer anderen Gruppe zusammen und führen Sie gegenseitig ein Review durch. Prüfen Sie, ob die Use-Case-Spezifikation den in der Aufgabenstellung aufgeführten Angaben entspricht. Verwenden Sie dabei die bereitgestellten Review-Vorlagen.

Ergebnis

Sie haben genau verstanden und dokumentiert, was die Aufgabe des CocktailPro ist. Unklarheiten haben Sie herausgearbeitet und durch Rückfragen geklärt. Die Qualität der Anforderungen wurde durch ein Review bestätigt.

---

[1] Sie sind in der Lage die Begriffe zu erklären, wenn der Betreuer Sie dazu befragt.

[2] Die Lieferung enthält eine h-Datei,.o-Datei, csv-Datei und main.cpp als Anwendungsbeispiel. Die csv-Datei enthält die Rezepte und muss im Ausführungsverzeichnis liegen. Wird die csv-Datei nicht gefunden, enthält das Rezeptbuch nur 5 Dummy-Rezepte.

Ihr Modell enthält ein Use Case Diagramm und mehrere sinnvolle und konsistente Use Cases. Die Spezifikationen sind ausgefüllt. Das zugelieferte Rezeptbuch ist in Ihr Modell eingebunden.

2. Termin: Design – der erste Entwurf

Vorbereitung

- Arbeiten Sie die Ergebnisse aus dem Anforderungsreview in Ihre Use Cases ein.
- Analysieren Sie Ihre Use Cases und leiten Sie daraus Analyseklassen ab (noch keine Methoden und Navigationsrichtungen). Zeichnen Sie ein entsprechendes Klassendiagramm auf einem Blatt Papier.
- Frischen Sie Ihr Wissen auf, über UML Klassendiagramme und das Suchen und Finden von Klassen (siehe auch Kapitel "Gutes Design" von OOAD) und über UML-Klassendiagramme
- Analysieren Sie das zugelieferte Rezeptbuch und das Beispiel bis Sie die Funktionsweise verstehen
- Lesen Sie in der Anleitung zu MagicDraw den Abschnitt zu Klassendiagrammen. Schauen Sie sich genau an, wie man Datentypen anlegt (Type, Type Modifier und Container).

2.1.) Klassendiagramm

Entwerfen Sie die erste Version des Klassenmodells indem Sie Ihre Use Cases analysieren und daraus Klassen ableiten. Legen Sie zuerst nur die Klassen an und wählen Sie aussagekräftige Namen (ohne Sonderzeichen und Umlaute). Anschließend dokumentieren Sie die Verantwortlichkeiten der Klassen[3]. Wenn die Klassenaufteilung stabil ist, ergänzen Sie Attribute (aber noch keine Operationen), Assoziation und Aggregationen/ Kompositionen. Prüfen Sie, ob es Generalisierungshierarchien gibt (welche Klassen haben gleiches Verhalten?). Beachten Sie die Regeln für einen guten Entwurf. Achten Sie dabei auf starken Zusammenhalt jeder Klasse und lose Kopplung der Klassen untereinander.

2.2.) Design eines "Mischbaren Rezeptbuchs"

In Ihrem Klassendiagramm taucht das Rezeptbuch auf. Eventuell haben Sie sogar das folgende "Problem" bereits festgestellt:
Das zugelieferte Rezeptbuch enthält viele Rezepte, von denen Sie aber nur wenige mischen können – je nachdem welche Zutaten zur Verfügung stehen. Überlegen Sie sich, welche Teile des Systems überhaupt von der Existenz dieser nicht-mischbaren Rezepte wissen müssen und kapseln Sie dieses Wissen in einem "Mischbaren Rezeptbuch", das nur noch die (mit den aktuellen Zutaten) mischbaren Rezepte anbietet.

2.3.) Verfeinerung des Designs

[3] Verwenden Sie in MagicDraw die Eigenschaft "Documentation" für die jeweiligen Klassen.

Erarbeiten Sie zu einigen Klassen (jedenfalls für das mischbare Rezeptbuch) die erforderlichen Methoden. Überlegen Sie sich, welche Klasse(n) in Ihrem System die mischbaren Rezepte bestimmen und welche Methoden Sie dafür einsetzen werden. Verfeinern Sie den Grobentwurf dieses Systemteils bis zu einem implementierbaren Design. Die Schnittstelle des "Mischbaren Rezeptbuchs" kann genau der Schnittstelle des zugelieferten Rezeptbuchs entsprechen. Erzeugen Sie bei Bedarf Konstruktoren, Destruktoren, Getter/Setter und verwenden Sie auch private-Methoden.

Ergebnis:
Ein Klassendiagramm beschreibt grob die Aufteilung des Systems in Klassen und Verantwortlichkeiten. Das Modell enthält Klassen mit ersten Methoden.

3. Termin: Design

Vorbereitung

- Frischen Sie Ihr Wissen auf, über UML-Sequenzdiagramme und Roundtrip-Engineering.
- Lesen Sie in der Anleitung zu MagicDraw die Abschnitte zu Sequenzdiagrammen und Code Generierung.
- Erarbeiten Sie ein Sequenzdiagramm das zeigt, wie die Initialisierung des Systems erfolgt[4].

3.1.) Verfeinerung des Entwurfs

Verfeinern Sie Ihre Klassen so, dass Sie die Methoden zur Umsetzung der Anwendungsfälle beinhalten. Benennen Sie die Objekte sinnvoll (z.B. mit vorangestellten der/die/das oder ein/eine). Beginnen Sie mit den Klassen für das „mischbare Rezeptbuch" und sorgen Sie dafür, dass diese Klassen möglichst vollständig sind.
Tipp: Investieren Sie besser hier etwas mehr Arbeit, bis Ihnen die Abläufe klar sind. Den Aufwand sparen Sie bei der Implementierung ein! Sie dürfen die Abläufe gerne als Sequenzdiagramme dokumentieren.

3.2.) Erster Implementierungszyklus

Das "mischbare Rezeptbuch" soll als erster Implementierungszyklus entworfen und umgesetzt werden.
Der Betreuer hilft Ihnen bei der Erzeugung der Coderahmen. Dabei achten Sie auf Meldungen, die auf Zyklen hinweisen. Falls eine solche Meldung erscheint, rufen Sie unbedingt einen Betreuer. Kompilieren Sie die leeren Coderahmen (Beachten Sie die Hinweise im Anhang A auf Seite 16.). Wandeln Sie die zugelieferte main.cpp so ab, dass Ihre neue Klasse verwendet wird. Kompilieren Sie alle Coderahmen zusammen mit der Main-Funktion und linken Sie das zugelieferte Rezeptbuch dazu (siehe auch Anhang A).

[4] Beachten Sie, dass Konstruktor-Aufrufe in Sequenzdiagrammen durch Pfeile auf das Objekt dargestellt werden.

Beginnen Sie noch nicht mit der eigentlichen Implementierung, sondern machen Sie lediglich erforderliche Anpassungen, bis der Code kompiliert. Importieren Sie anschließend den Code durch Reverse-Engineering, erzeugen Sie den Code neu und überprüfen Sie noch einmal die Kompilierbarkeit.

3.3.) Erste Implementierung

Entwickeln Sie bis zum nächsten Termin den Code für das "Mischbare Rezeptbuch"[5]. Dabei wird die Zutatendatei eingelesen und die mischbaren Rezepte werden bestimmt.

Ergebnis

Der Entwurf ist soweit verfeinert, dass die Methoden zur Umsetzung definiert sind. Die Coderahmen für das "Mischbare Rezeptbuch" sind generiert und kompilierbar.
Ihnen ist klar, wie die Umsetzung der Funktionalität mit den geforderten Methoden funktioniert.

4. Termin: Vorbereitung der restlichen Implementierung

Vorbereitung

- Implementieren Sie das "Mischbare Rezeptbuch" und testen Sie dessen Funktion inklusive des Einlesens der (zugelieferten) Zutatenliste. Ihr Programm zeigt an, welche der Rezepte mit den gegebenen Zutaten mischbar sind. Überprüfen Sie Ihr Ergebnis, indem Sie die Rezepte und die Zutaten (beides Textdateien) analysieren.
- Frischen Sie Ihr Wissen über das Beobachter-Muster auf und überlegen Sie sich mit einem Sequenzdiagramm wie Sie das Beobachter-Muster anwenden müssen.

4.1.) Reverse-Engineering

Zuerst muss der Code, den Sie für das mischbare Rezeptbuch implementiert haben, in das Modell importiert werden. Dazu führen Sie ein Reverse-Engineering dieser Klassen durch. Der Betreuer hilft Ihnen dabei.

4.2.) Beobachter-Muster

Identifizieren Sie die zur Anwendung des Beobachter-Musters notwendigen Klassen. Falls noch Klassen fehlen, legen Sie diese bitte (ohne jegliche Details) an. Anschließend werden die Klassen durch MagicDraw mit den entsprechenden Methoden versehen.
Legen Sie ein neues Klassendiagramm an und stellen Sie die Klassen der Observer-Musters so dar, dass das Beobachtermuster als solches erkennbar ist

[5] Bei seltsamen Effekten mit Stringvergleichen beachten Sie bitte den Hinweis zu den unterschiedlichen Dateiformaten von Windows und Unix. Beachten Sie auch die zugelieferten Codebeispiele (z.B. zum Einlesen einer Textdatei).

Zeichnen Sie unter Berücksichtigung des Beobachter-Muster 1-2 Sequenzdiagramme, die zeigen, wie beim Dosieren die Interaktion zwischen einem Dosierer, dem Display und der Waage funktioniert.
Zeigen Sie, wie bei der Dosierung die Gewichtserhöhung bei der Waage simuliert wird.

4.3.) Aktualisierung des Roundtrip-Engineering

Bevor Sie diesen Punkt angehen, sollte Ihr Modell konsistent und vollständig sein. Falls Sie mit einem unvollständigen oder fehlerhaften Modell weitermachen, wird das später erheblichen Zusatzaufwand verursachen! Ergänzen Sie Ihr Codeenineering-Set um die neu angelegten Klassen für das Beobachter-Muster.
Erzeugen Sie durch Forward Engineering die Coderahmen. Passen Sie den Code so an, dass er wieder kompiliert. Falls Magicdraw dabei eine Meldung wegen zyklischer Abhängigkeiten bringt, rufen Sie einen Betreuer, der Ihnen hilft den Zyklus aufzulösen. Anschließend importieren Sie den Code wieder in das Modell (Reverse) und erzeugen anschließend wieder die Coderahmen aus dem Modell. Wenn der Code dann immer noch kompiliert, funktioniert das Roundtrip-Engineering.

4.4.) Implementierung

Entwickeln Sie **bis zum Beginn (!!!) des nächsten Termins** den Rest des Systems. Halten Sie sich bei der Implementierung an die Entwurfsentscheidungen in Ihren Klassen- und Sequenzdiagrammen.
Tipps:

- Lesen Sie die Hinweise auf Seite 8.
- Nutzen Sie die Zerlegung Ihrer Systems in Klassen zur Aufteilung der Implementierung innerhalb Ihres Teams.
- Achten Sie bei der Implementierung darauf, dass der Code und das Modell konsistent bleiben. Änderungen an der Signatur von Klassen/Methoden nehmen Sie am besten nur im Code vor. Das Ergebnis können Sie dann im nächsten Praktikum in Ihr Modell importieren (Reverse Engineering).
- Bei größeren Problemen bei der Implementierung ist oft ein falsches Design die Ursache. Es ist dann geschickter, einen Schritt zurück zu gehen und das Design zu überarbeiten. Falls Sie schon Code geschrieben haben, müssen Sie aber zuerst den Code importieren bevor Sie Änderungen im Modell machen können.

Ergebnis

Ein vollständiges Design, das die zu implementierenden Klassen und Methoden beschreibt und mit Hilfe von Sequenzdiagrammen "getestet" wurde und. Leere Coderahmen, die aber schon kompilierbar sind.

5. Termin: Abgabe des Produkts beim Kunden und Dokumentation

Dieser Termin entspricht dem Abgabetermin bei Ihrem Auftraggeber.

Vorbereitung

Zu **Beginn** des Termins muss Ihr Code bereits vollständig und lauffähig sein.
Achtung! Sollte Ihr Code unvollständig oder offensichtlich nicht lauffähig sein, so erhalten Sie sofort eine rote Karte und sind durchgefallen. Das ist sehr streng? Dann überlegen Sie einmal was Ihr Auftraggeber mit Ihnen machen würde, wenn Sie am Abgabetermin ohne brauchbares Ergebnis auftauchen…
Sie sollten Sie also lieber nicht bis zum letzten Tag mit der Lösung der Aufgabe warten.

5.1.) Importieren des Codes

Mit Reverse-Engineering werden Ihr Code und eventuelle Änderungen an Klassen oder Methoden in Ihr Modell übernommen. Fehler in diesem Arbeitsschritt können leicht Ihr Modell zerstören. Fragen Sie deshalb im Zweifelsfall Ihren Betreuer anstatt herum zu probieren.

5.2.) Zusammenstellung der Dokumentation

Erzeugen Sie eine Dokumentation mit MagicDraw und speichern Sie das Ergebnis als PDF-Datei.

5.3.) Endabnahme durch den Anwender

Die Endabnahme wird durch den Auftraggeber durchgeführt. Zur Abnahme gehört auch das Prüfen des Modells mit Dokumentation. Prüfen Sie vorher selbst, ob Ihr System die Abnahmetests, die den Use Cases aus der ersten Übung entsprechen, besteht.

Ergebnis

Ein wartbares System, das die Anforderungen erfüllt und den Abnahmetest bestanden hat.

5.4.) Qualitätssicherung

Nach bestandener Abnahme führen Sie Schritte zur Qualitätssicherung durch. Diese finden Sie in dem Dokument Qualitätssicherung. Sie können bereits ab dem Zeitpunkt Qualitätssicherung betreiben, wenn Sie ihr erstes lauffähiges Programm haben (Termin 3).

Anhang A

Verwendung des Linkers

Nach dem Kompilieren des Sourcecodes wird der Linker aufgerufen, der die vom Compiler erzeugten Binärdateien zusammenfügt. Das heißt, die Aufrufe von Methoden und Klassen werden mit deren Binärcode verknüpft. Beim "normalen" Kompilieren von C++-Dateien erfolgt der Linkeraufruf automatisch und Sie bemerken den Linker kaum (bis auf gelegentliche unverständliche Fehlermeldungen). Für das zugelieferte Rezeptbuch müssen Sie allerdings die entsprechende Binärdatei manuell zum Linkvorgang hinzufügen. Wie das für die gängigsten Compiler/IDEs geht, steht hier:

Das Rezeptbuch wird (auf der Homepage) zur Verfügung gestellt

- als Binärdatei (.o, .obj) für verschiedene OSs, CPUs und Compiler
  Sie müssen natürlich die Datei verwenden, die zu Ihrem Rechner passt – also unterschiedliche Dateien auf Linux, PC, MAC – und dann auch noch abhängig von Ihrem Compiler.
  Verwenden Sie die falsche Datei, gibt es üble Fehlermeldungen vom Linker.
- mit einer Header-Datei (.h) und
- einem Anwendungsbeispiel (main.cpp)
- Der Source-Code ist nicht zugänglich

a) GNU-Compiler (g++)

- Aufruf: g++ main.cpp -o Rezeptbuch.exe RezeptbuchXXX.o

b) Microsoft Visual Studio :

- Das passende Binary (Rezeptbuch…VSxx.obj) mit Drag & Drop in den Ordner "Ressourcendateien" in die Projektmappe werfen.

c) Code::Blocks:

- Das passende Binary in den „Build Options" des Projekts unter „Linker settings" als „Link library" eintragen.

d) Netbeans:

- Wählen Sie mit der rechten Maustaste Resource Files -> Add Existing Item und wählen Sie das Binary aus.

- Nun müssen die Kompiliereinstellung geändert werden. Hierzu wieder ein Rechtsklick auf das eingefügte Objekt und Eigenschaften wählen. Hier kann nun unter Tool "C++ Compiler" ausgewählt werden.
- Zu guter Letzt muss die Objektdatei noch in den Ordner build/Debug/GNU-Linux-x86/ kopiert werden.