UML Dokumentation

Damien Flury

27. Februar 2019

Inhaltsverzeichnis

```
public class Person {
    private final String firstName;
    private final String lastName;
    public Person(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }
}

public final class Employee extends Person {
    private final String teamName;
    public Employee(String firstName, String lastName, string teamName) {
        super(firstName, lastName);
        this.teamName = teamName;
    }
}
```

Abbildung 1: Vererbung in Java

1 Einführung

2 Klassendiagramme

2.1 Vererbung

Bei der Vererbung (engl. Inheritance) von Klassen werden Unterklassen generalisiert. Java verfügt im Gegensatz zu zum Beispiel C++ nur über Single Inheritance, das heisst, dass eine Klasse jeweils nur von maximal einer anderen Basisklasse erben kann. Dies führt zu einigen Limitierungen, vereinfacht den Code jedoch. Somit muss in Java gegebenenfalls auf andere Beziehungen ausgewichen werden, wie zum Beispiel die Komposition. Codebeispiel: Abbildung 1, grafische Beispiele: Abbildung 2, Abbildung 3 und Abbildung 4.

2.2 Assoziationen

Assoziationen bezeichnen sowohl Besitz- als auch Kennbeziehungen. Dies funktioniert im Beispiel von Java häufig anhand des Konstruktors oder mithilfe von Setter-Methoden. Codebeispiel: Abbildung 5, UML-Diagramme sind unter Abbildung 6 und Abbildung 7 zu finden.

3 Use Case Diagram

Das Use Case Diagram bietet die Möglichkeit, Benutzerinteraktion mit der Applikation zu repräsentieren. Zum ersten Mal vorgestellt wurde es an einer Konferenz in 1987 von

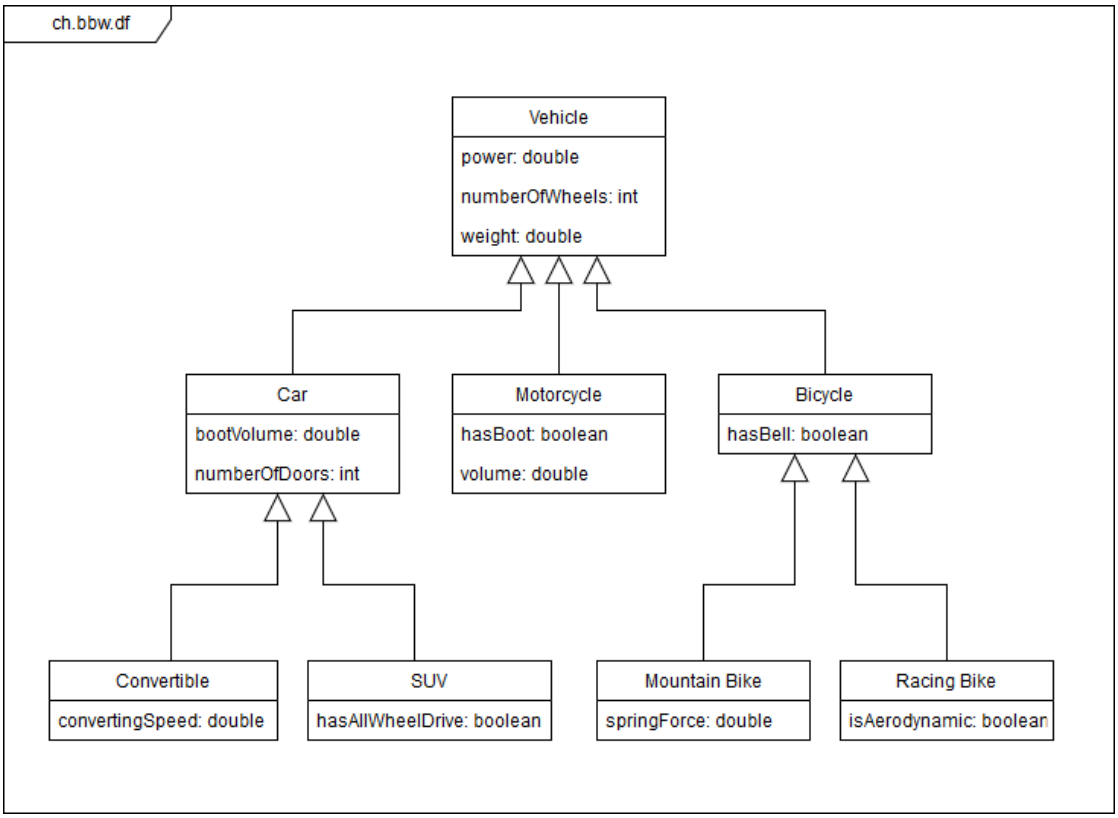

Abbildung 2: UML-Diagramm mit Vererbung (draw.io)

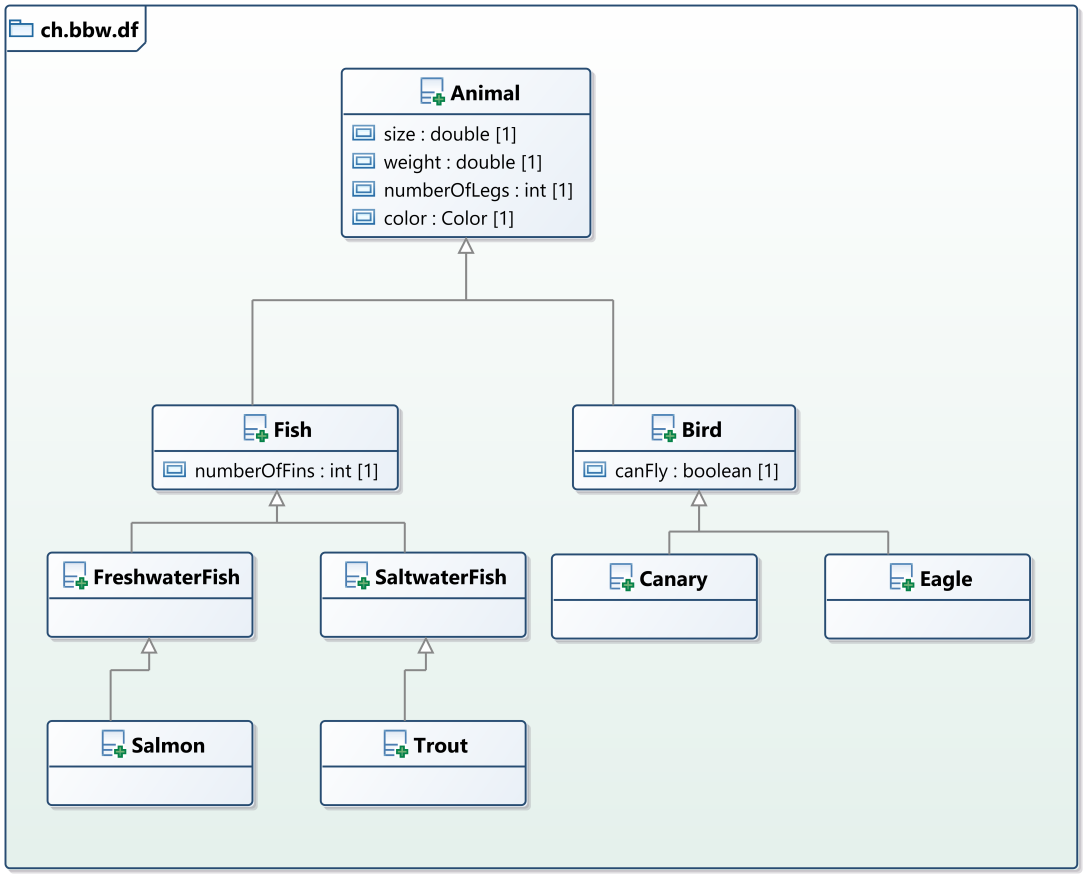

Abbildung 3: UML-Diagramm mit Vererbung (UML Designer)

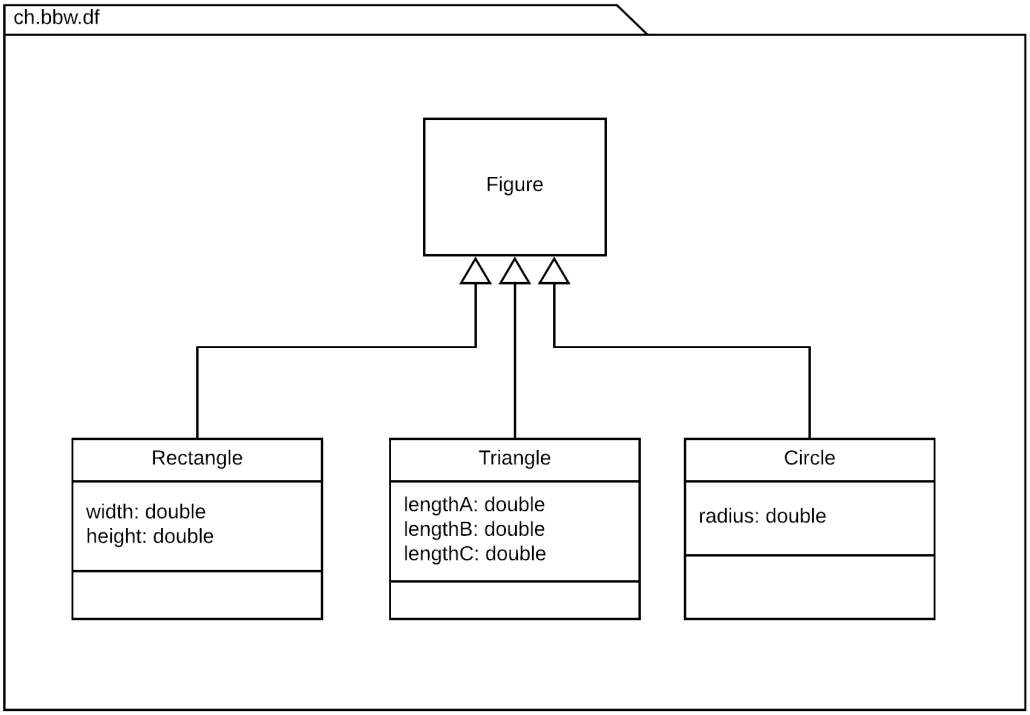

Abbildung 4: Klassendiagramm mit Vererbung (Lucidchart)

```java
public final class Application {
    public static void main(String[] args) {
        var bike = new Bike();

        // Hier wird dem Driver ein Bike uebergeben:
        var driver = new Driver(bike);
    }
}

public final class Driver {
    private final Bike bike;
    public Driver(Bike bike) {
        this.bike = bike;
    }
}
```

Abbildung 5: Aggregation in Java

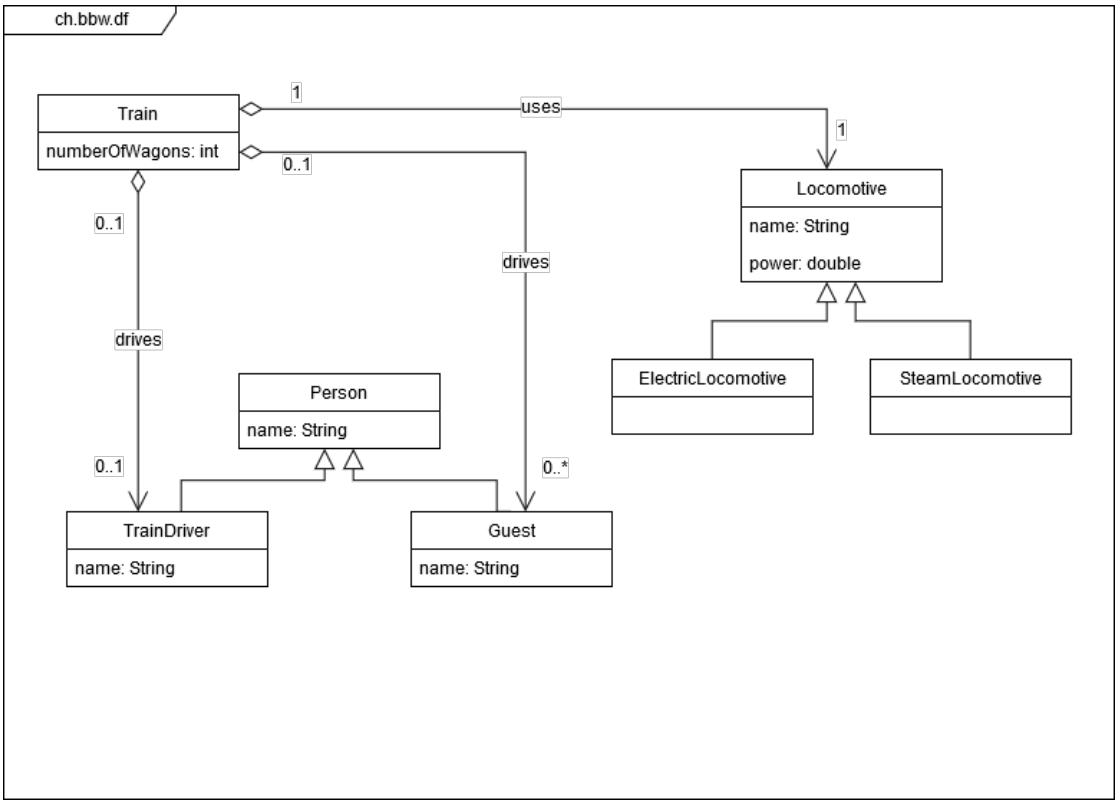

Abbildung 6: Klassendiagramm mit Assoziationen (draw.io)

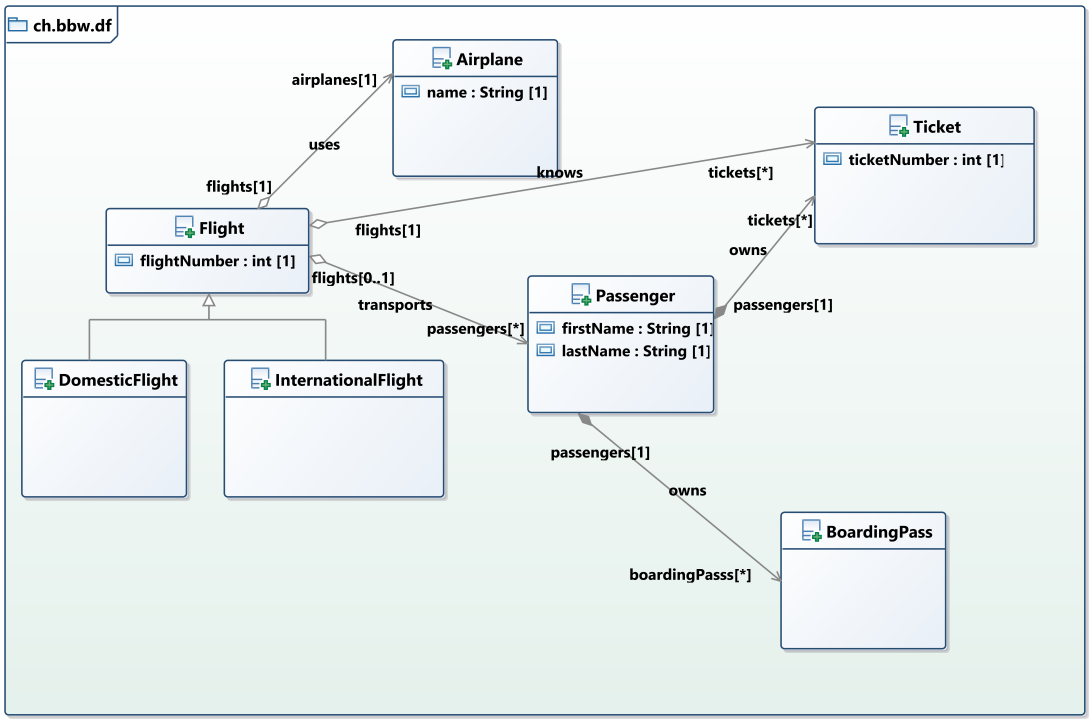

Abbildung 7: Klassendiagramm mit Assoziationen (UML Designer)

Jacobson und findet heutzutage häufig einen Platz in unterschiedlichen Softwareteams mit diversen Programmiersprachen. Beispiele: Abbildung 8 und Abbildung 9

4 Aktivitätsdiagramm

Mithilfe des Aktivitätsdiagramms kann man Abläufe, Aktionen und Kontrollflüsse beschreiben.

Häufig werden durch Aktivitätsdiagramme einzelne Use Cases genauer beschrieben. Das Aktivitätsdiagramm ermöglicht hier die Darstellung von ein wenig komplexeren Abläufen mit Falls-Verzweigungen, Wiederholungen und anderen Ausnahmen. Beispiele: Abbildung 10 und Abbildung 11

5 Fazit zu den Tools

5.1 Draw.io

Draw.io ermöglicht das Zeichnen von vielen verschiedenen Diagrammen und hat meine Bedürfnisse für alle Aufgabenstellung erfüllt. Die Benutzeroberfläche bietet viele Möglichkeiten und wird nach einer Zeit ziemlich intuitiv und einfach.

Ausserdem ist Draw.io eine Webapplikation, benötigt also keinen Download und hat direkte Unterstützung von Google Drive.

Draw.io ist mit Abstand mein Lieblingstool, da es viele Möglichkeiten benutzerfreundlich bereitstellt.

5.2 Visio 2013

Visio bietet viele Möglichkeiten, ist jedoch meiner Meinung nach nicht sehr benutzerfreundlich. Ausserdem benötigt die Software eine teure Lizenz und läuft nur auf Microsoft Windows. Das Endprodukt finde ich persönlich eher unbefriedigend und qualitativ nicht so hochwertig wie das von anderen Tools.

5.3 UML Designer

UML Designer bietet eine hohe Variation von Möglichkeiten, ist jedoch besonders am Anfang sehr kompliziert. Ausserdem ist alles sehr Java- und Eclipsebasiert, somit denke ich, dass UML Designer für andere Sprachen und Entwicklungsumgebungen nicht die beste Lösung ist.

Das Endprodukt finde ich besonders ansprechend und gewinnt deshalb unter allen Tools.

5.4 UMLet

UMLet bietet sehr wenige Möglichkeiten von Anfang an. Während Klassendiagramme relativ einfach umzusetzen sind, sind Aktivitätsdiagramme sehr aufwendig (aber nicht

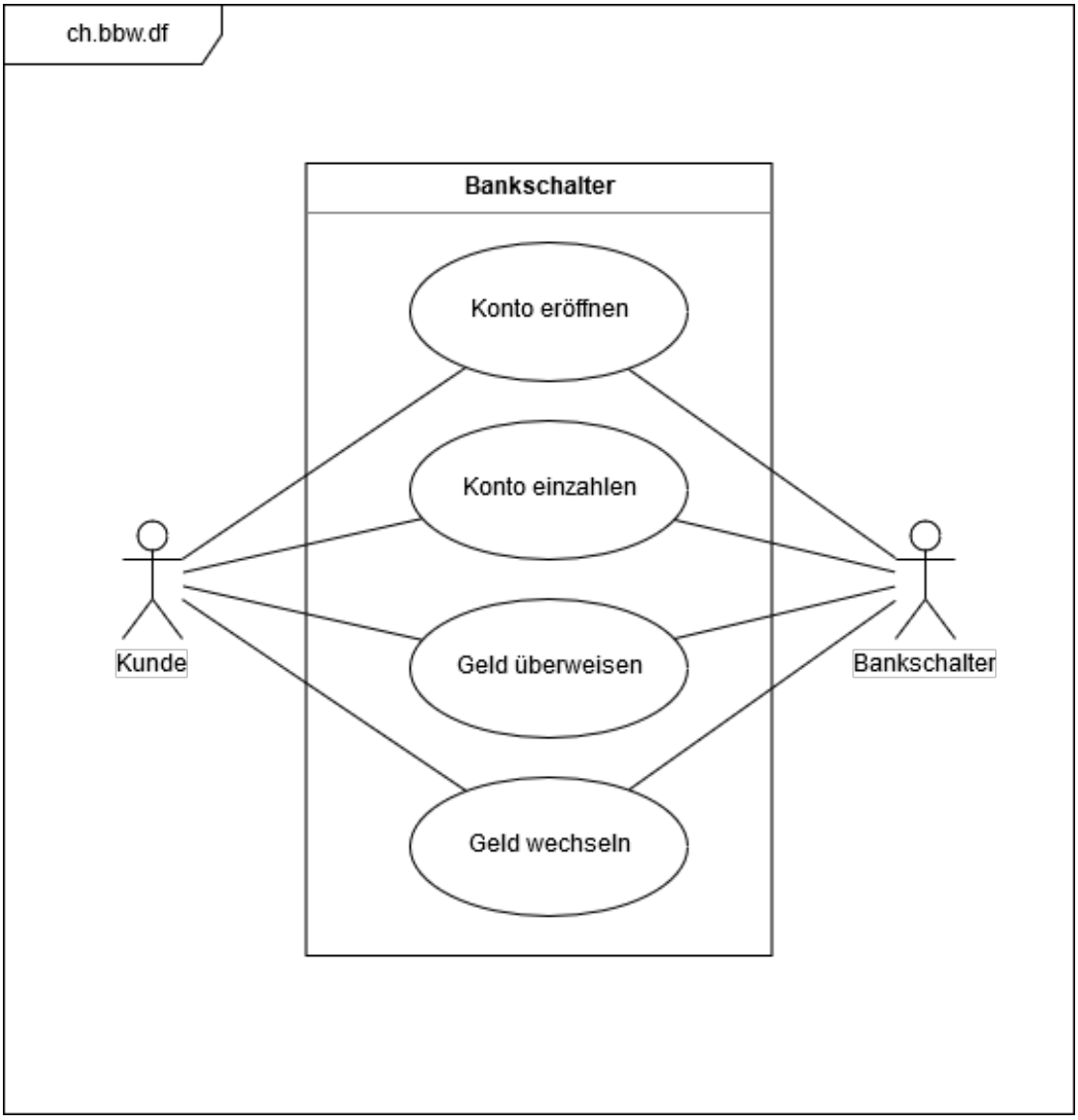

Abbildung 8: Use Case Diagram (draw.io)

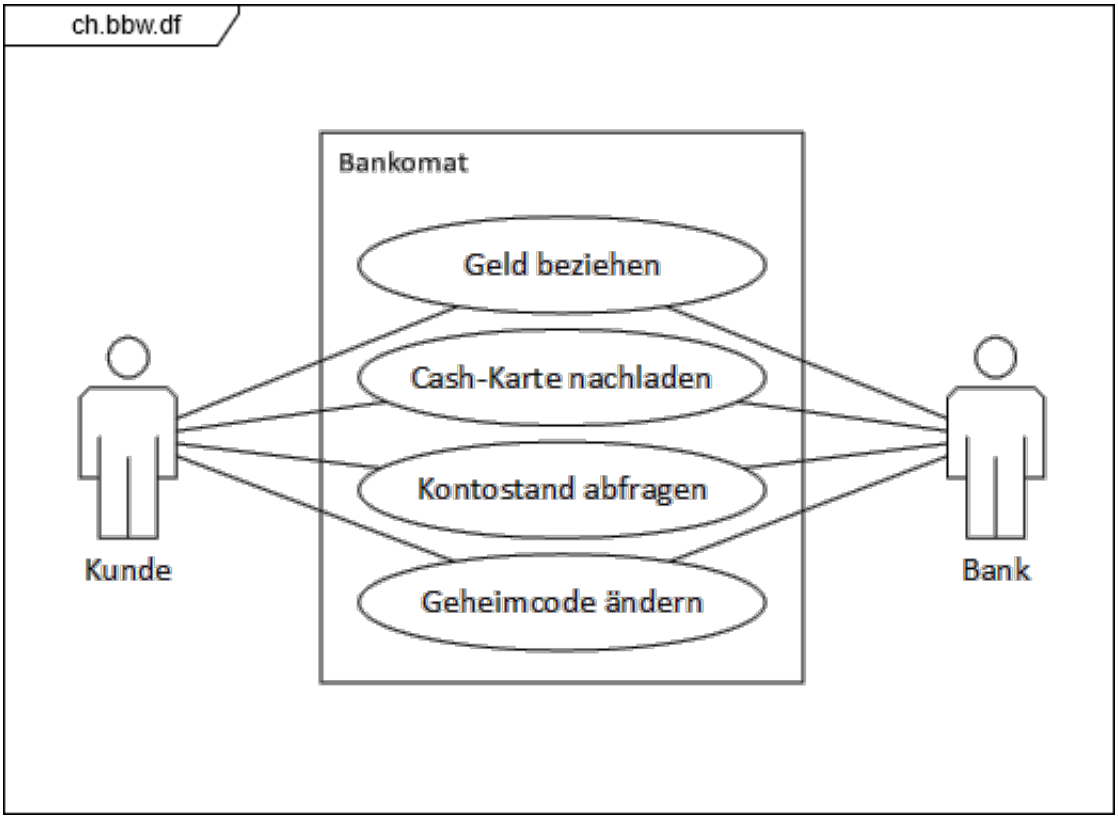

Abbildung 9: Use Case Diagram (Microsoft Visio)

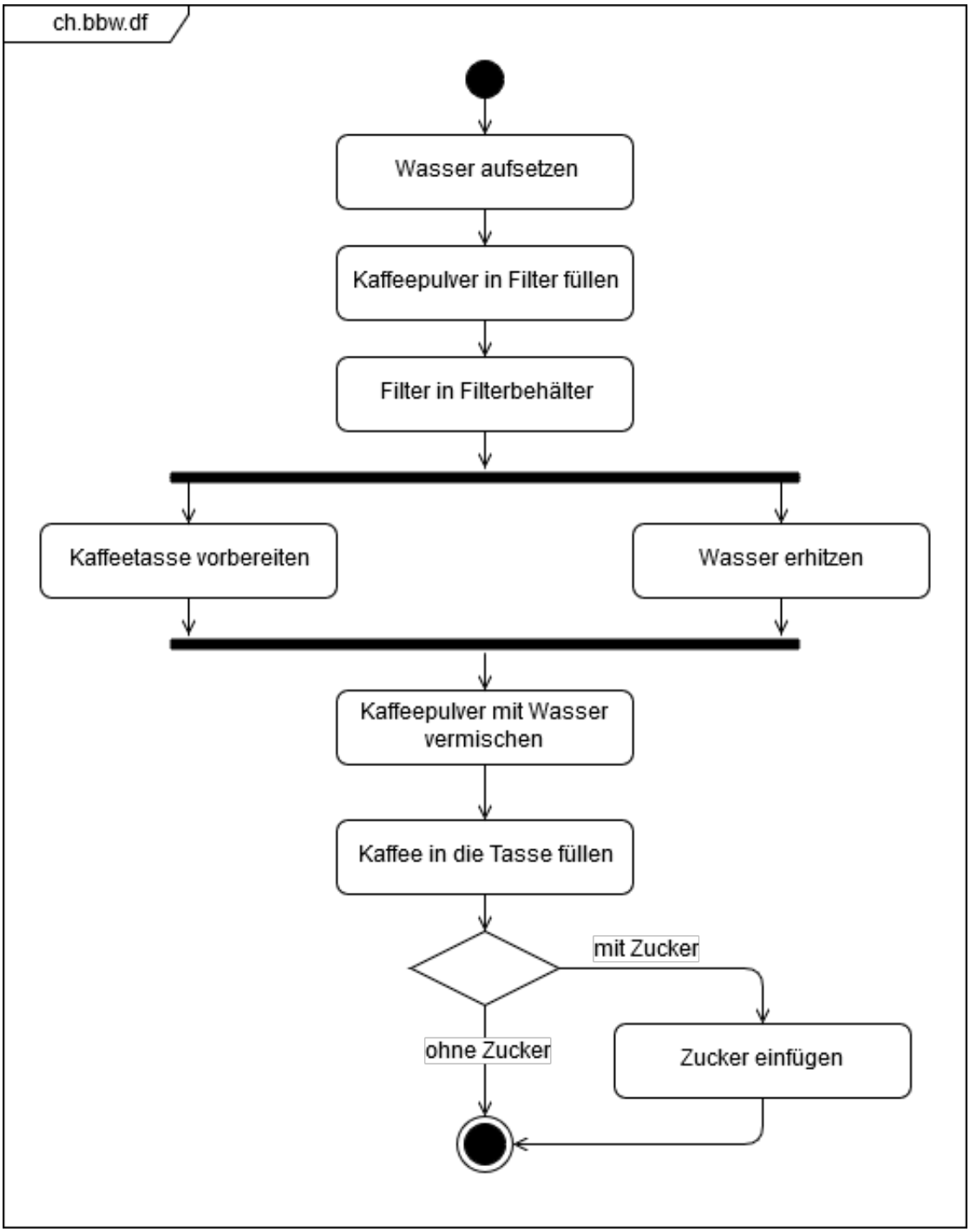

Abbildung 10: Aktivitätsdiagramm (draw.io)

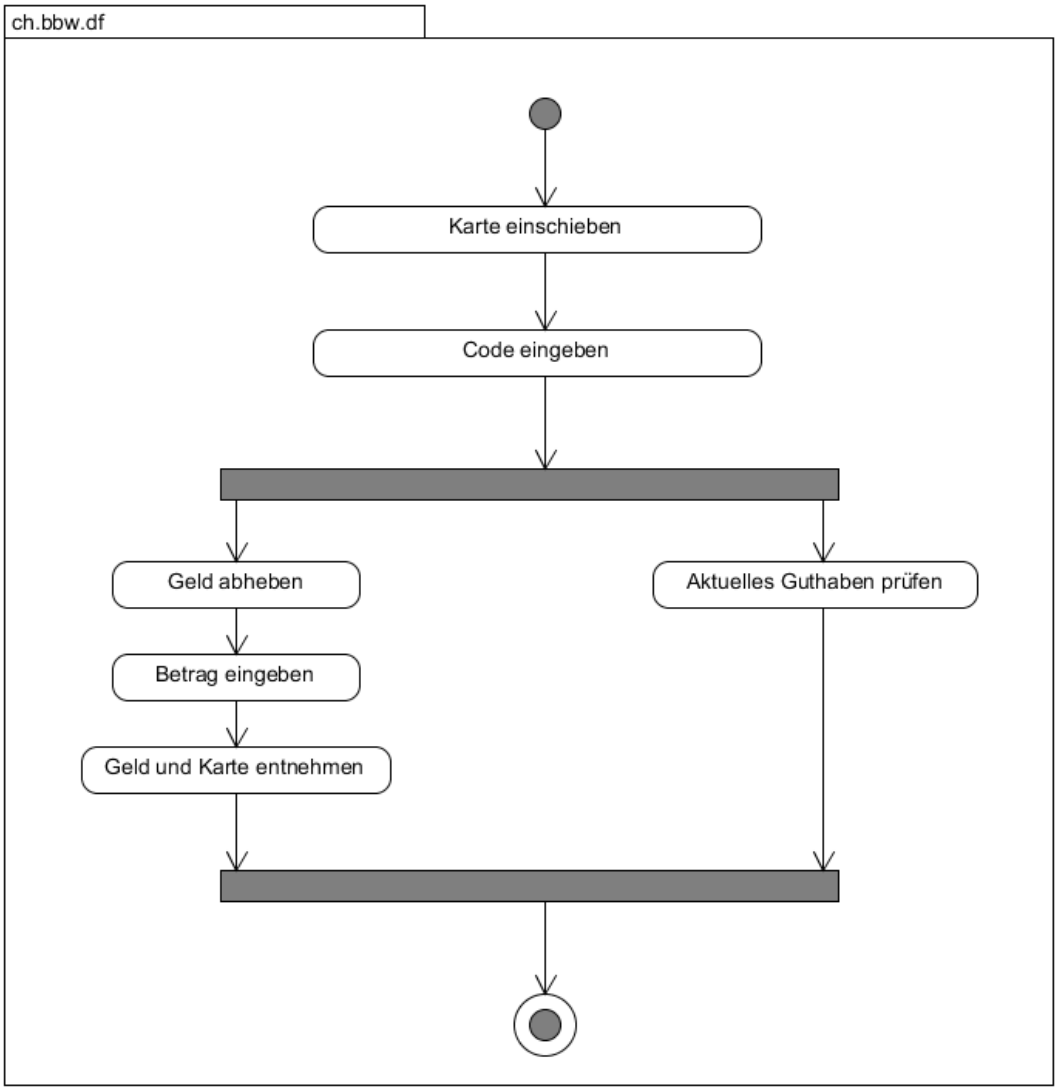

Abbildung 11: Aktivitätsdiagramm (UMLet)

unmöglich). Ich war mir dies leider im Voraus nicht bewusst und habe somit ein Aktivitätsdiagramm mithilfe von UMLet erstellt. Das integrierte Scripting-Tool ist sehr mächtig und erlaubt die Erzeugung von komplexeren Formen, ist jedoch kompliziert und man findet wenig Dokumentation im Internet.

5.5 Lucidchart

Lucidchart ist ziemlich ähnlich zu Draw.io. Es bietet einen riesigen Umfang an Funktionen, ich finde es jedoch ein wenig komplizierter als Draw.io. Ansonsten ist die Benutzeroberfläche ziemlich gut und angenehm zu verwenden.

6 Reflexion

Ich habe normalerweise zuerst die Diagramme gezeichnet und diese danach mit ein wenig Text versehen. Geklärt hat sich hauptsächlich weiter der Unterschied zwischen Komposition und Aggregation. Ausserdem konnte ich Einblicke in unterschiedliche Diagrammtypen und Zeichnungstools gewinnen.

Schwer fiel mir vorallem die Angewöhnung an die verschiedenen Tools. Draw.io kannte ich bereits zum Teil, alle anderen waren für mich neu und unbekannt.

Grundsätzlich bin ich sehr zufrieden mit meiner Arbeit, ich habe in sinnvollen zeitlichen Etappen gearbeitet und war kaum im Verzug. Ich hätte mich besser mit den Tools vertraut machen können, bevor ich die Diagramme gezeichnet habe, so wäre mir vermutlich auch aufgefallen, dass UMLet wesentlich besser für Klassendiagramme als für Anwendungsfalldiagramme geeignet ist.