

Übung 02

JULIA ZAMAITAT ■■■■■ LOTTE UNCKELL ■■■■■

REPORT

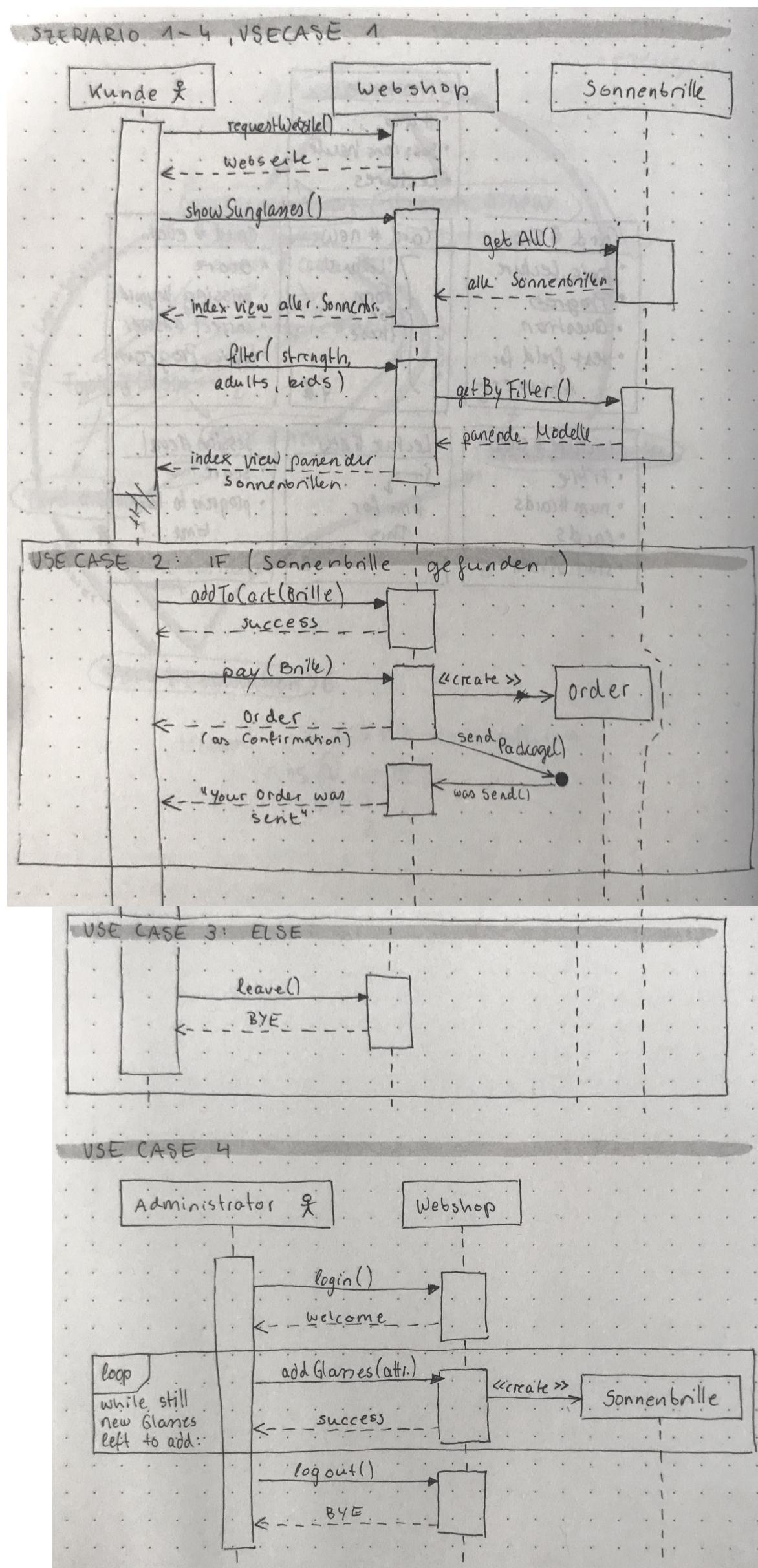
1. Sequenzdiagramme

In dieser Übung lernten wir zwei neue Diagramm-Arten kennen. Wir begannen mit den Sequenzdiagrammen. Mit Hilfe von Sequenzdiagrammen (im weiteren Verlauf SD genannt) kann man das Verhalten einer Software von innen beschreiben. Sie sind Interaktionsdiagramme zwischen verschiedenen Akteuren und Aktivitäten, also Methodenaufrufen. Sie bilden die einzelnen Schritte eines Szenarios visuell ab - dabei kann so ein SD je nach Anzahl der benötigten Schritte ziemlich lang werden!

Zu Beginn der Übung haben wir uns die Frage gestellt, warum man NOCH ein Diagramm braucht, um die Abläufe der Software darzustellen. Würden die Szenarios der Use Cases dafür nicht vielleicht ausreichen? Schlussendlich denken wir, dass die Szenarios eher für den groben Überblick gedacht und geeignet sind, und die SD nützlich werden, wenn man sich an die konkrete Implementierung einer Klasse machen will, da der "Bauplan" einfach viel genauer ist, und man quasi "nur noch" aus den beschreibenden Schritten den Code schreiben muss. Ein bisschen nervig ist jedoch, dass schon wieder andere Pfeile eingeführt werden und die Bedeutung dieser Pfeile sich je nach Diagramm unterscheidet – das ist der intuitiven Benutzung der Pfeile nun nicht wirklich zuträglich.

So viel zur Theorie – in der Praxis gestaltete sich das Erstellen des SD als einfach, da wir unsere Szenarien schon sehr detailliert beschrieben hatten. Es kam die Frage auf, wie man grafisch zu erkennen geben kann, wenn ein SD an ein anderes SD anknüpft, dass haben wir einfach mit if-else Blocks gelöst, sodass verschiedene Handlungswege gleich auf einem Blick zu erkennen sind. Insgesamt brauchten wir vielleicht 15 Minuten zum Skizzieren der SD und es sind keine Fragen offen geblieben.

Unser Sequenzdiagramm:



2. State Machine Diagrams

Das zweite Diagramm, was wir an diesem Tag kennengelernten, war das State Machine Diagram (im weiteren Verlauf SMD genannt). Obwohl - "kennenlernen" ist vielleicht der falsche Ausdruck - in Info2 behandelten wir die SMDs bereits, wenn auch mit einer etwas anderen Notationsform, obwohl das auch von Prof. zu Prof. variierte. Ein SMD beschreibt und vor allem visualisiert die verschiedenen Zustände eines Objekts und die Wege, die zu diesen Zuständen führen. Sie sind wichtig, um das Verhalten und das "Wissen" komplexer Objekte verstehen zu können - so zumindest laut Definition. So richtig hat sich für uns der Vorteil zur Benutzung an echten Programmen noch nicht ergeben, da auch noch nie "in real life" so verwendet. Klar, wir haben schon alle möglichen SMDs gezeichnet für verschiedene Automaten, aber was uns das programmiertechnisch gebracht hätte, ist uns unklar. Vielleicht findet sich in der Übung mal ein Anwendungsbeispiel?

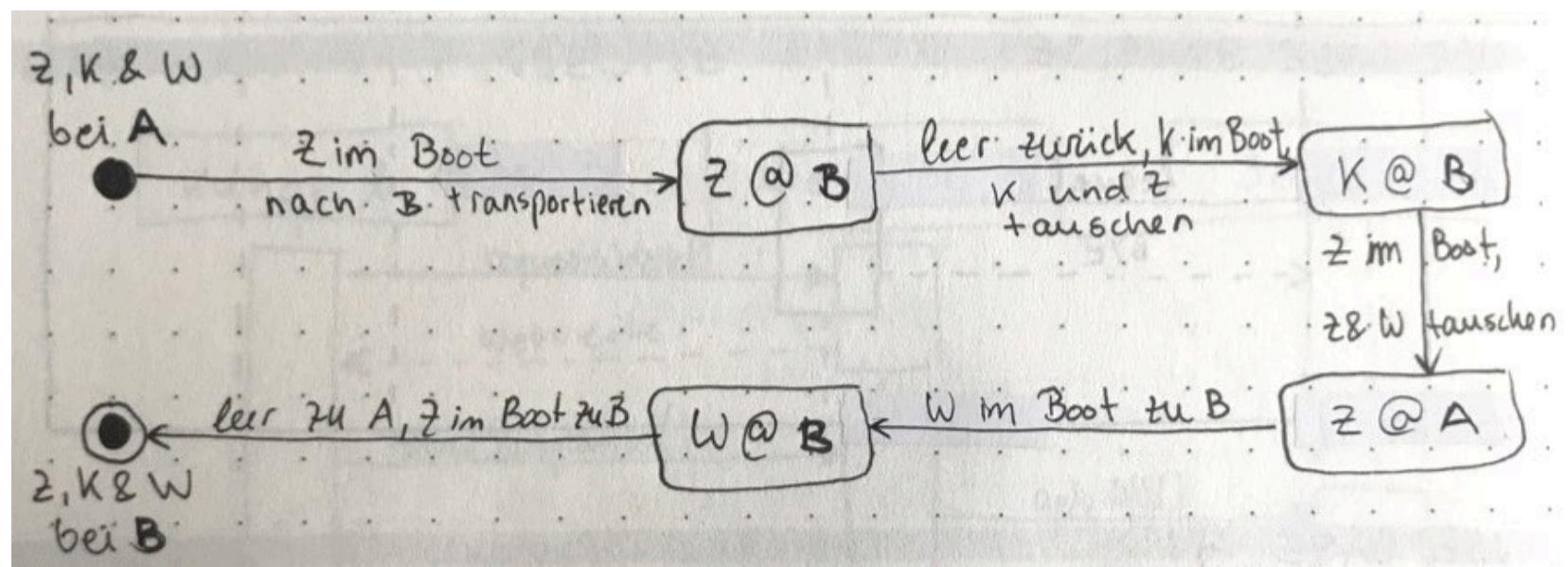
SMDs haben ihre eigene spezifische Notation (of course!) und arbeiten mit der Darstellung von States durch Pfeile, einem Initial State und einem Final State sowie Transitions und Events dazwischen. Ein State repräsentiert die Gegebenheiten eines Objekts zu einer bestimmten Zeit in seinem Lebenszyklus. Es gibt Events, die zu einer Transition von einem State in den anderen führen. Die Transitions kennzeichnet man mit Pfeilen, das dazugehörige Event wird darüber notiert. Wir haben noch etwas zu Actions und Activities gelesen, aber da wir die Unterscheidung in der Übung nicht vorgenommen haben, wissen wir nicht, ob es relevant ist, dazwischen genau differenzieren zu können. Allgemein zeigen verschiedene Internetquellen immer wieder neue andere Subkomponenten des SMD, sodass wir etwas verunsichert sind, welche Elemente wirklich alle benötigt werden und wie genau unser Wissen im Detailgrad sein muss.

Die Zeichnung der SMDs war nicht so einfach wie die der SDs, da wir uns am Anfang die Frage stellten, auf welche Objekte sich unsere States bezogen. Bei dem Wolf-Ziege-Kohl Problem hätten wir vom State von Ufer A oder Ufer B ausgehen können, entschieden uns aber, uns auf die States der Tiere/des Kohls zu beziehen.

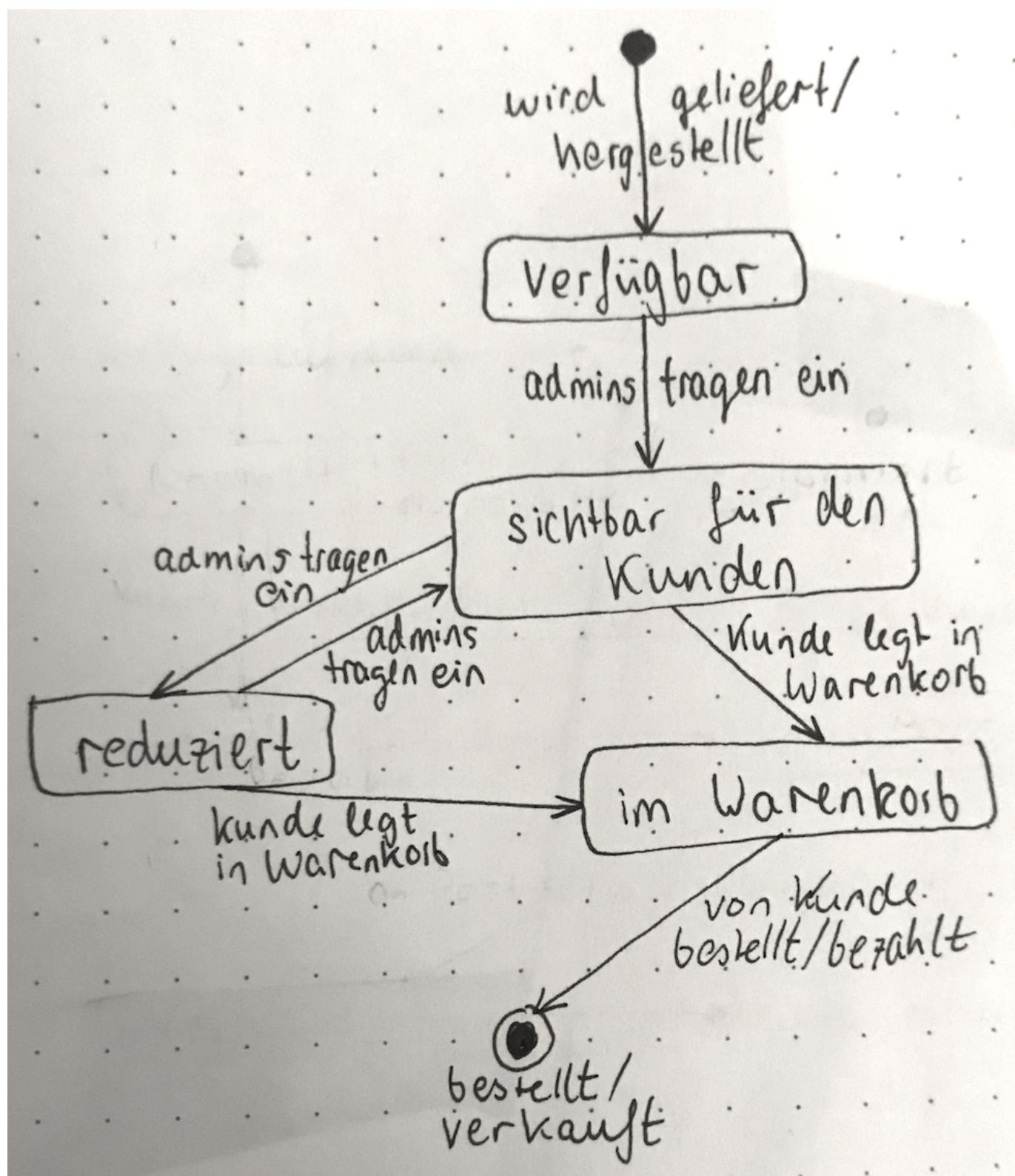
Für das SMD der Sonnenbrille war uns erst nicht klar, ob wir uns auf das reale Objekt Sonnenbrille oder auf die Klasse Sonnenbrille in der Software beziehen sollten. Auf Nachfrage entschieden wir uns für die Klasse in der Software.

Nachdem diese beiden Entscheidungen getroffen waren und wir die States und die Transitions für alle drei SMDs in Listenform aufgeschrieben hatten, war dann das Zeichnen der eigentlichen Diagramme nicht mehr sonderlich kompliziert. Die Ergebnisse finden sich auf der nächsten Seite.

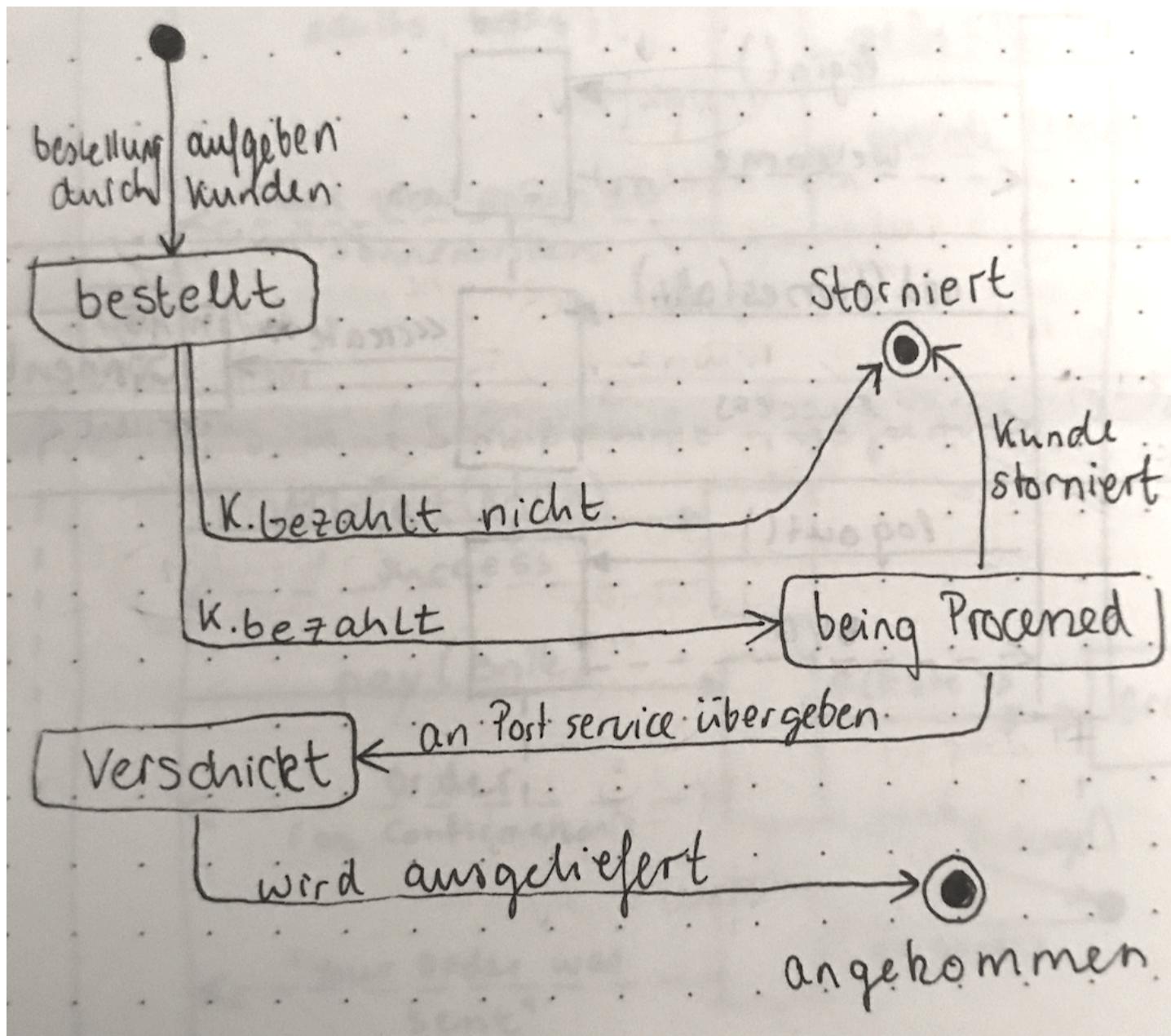
Unsere State Machine Diagrams:



Wolf, Ziege, Kohl - Problem



States einer Sonnenbrille im Shop



States einer Bestellung