Software Engineering I

Musterlösungen zur Klausur vom 13.8.2005

Aufgabe 1

a) Einen Mechanismus für den AF "Vergehen erfassen" zeigt die Abb. 1.

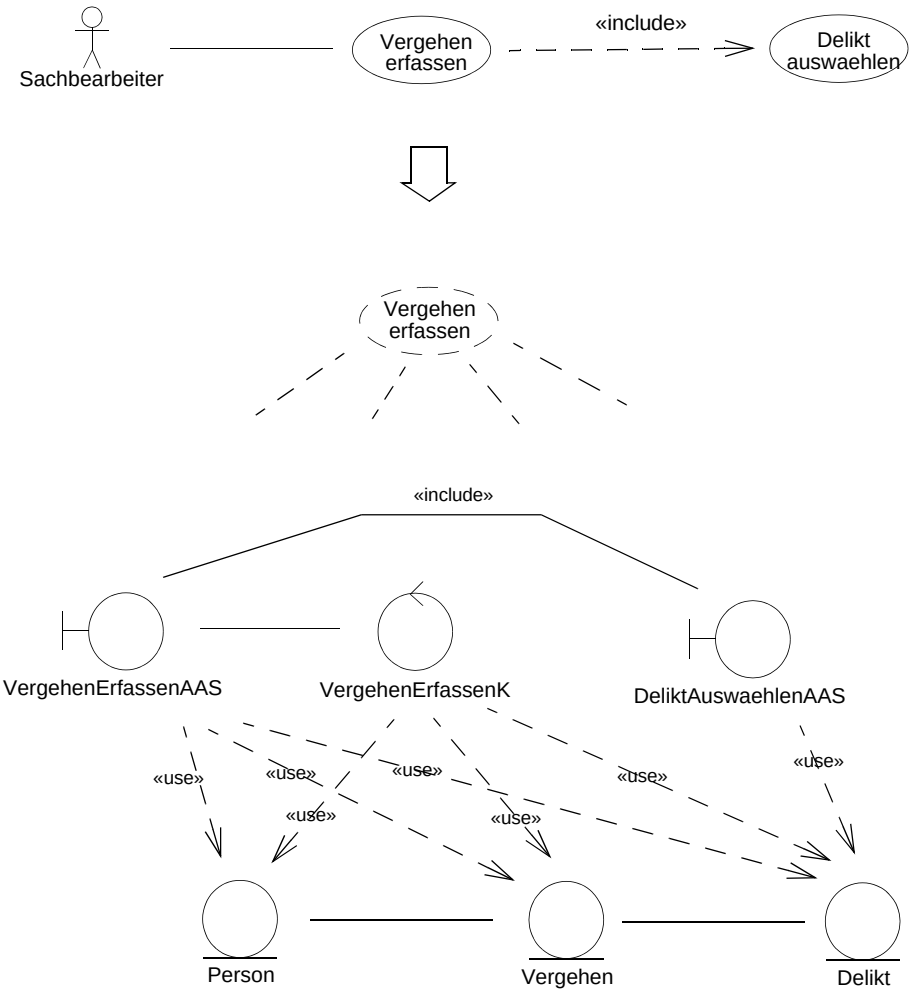

Abb. 1 Mechanismus für den Anwendungsfall "Vergehen erfassen"

b) Sequenzdiagramm: Wie in der Aufgabenstellung vorgegeben wird davon ausgegangen, dass die betroffene Person bei der Erzeugung der VergehenErfassenAAS-Instanz mit übergeben wird.

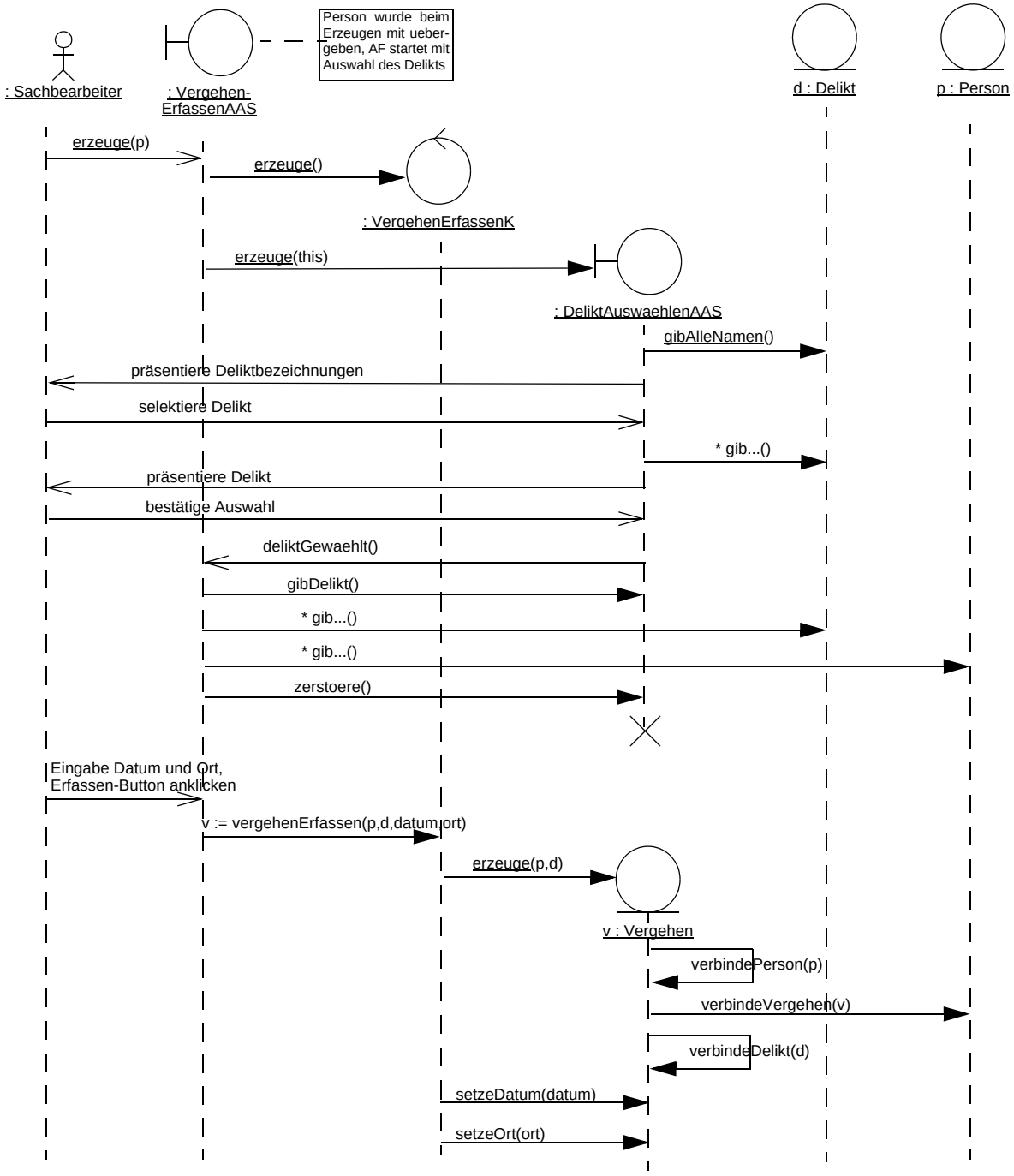

Abb. 2 Sequenzdiagramm für das Szenario “Vergehen erfassen - erfolgreich“

Bei dieser Aufgabe sind sehr unterschiedliche Lösungen denkbar, was bei der Bewertung natürlich entsprechend berücksichtigt wird.

c) Erweitertes Anwendungsfalldiagramm:

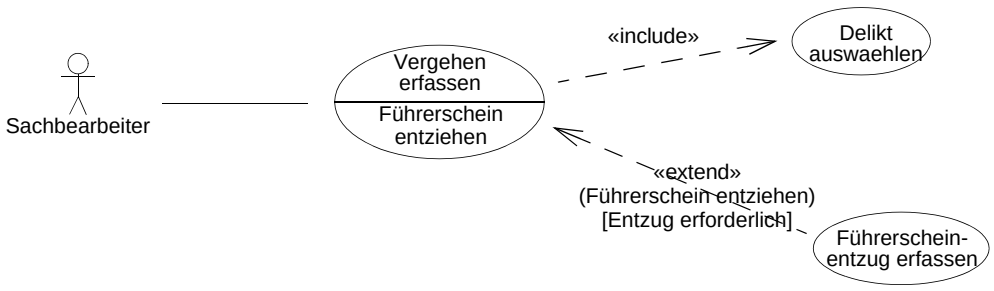

Abb. 3 Erweitertes Anwendungsfalldiagramm, Aufgabe 1d

Textuelle Spezifikation der Anwendungsfälle "Vergehen erfassen" und "Führerscheinentzug erfassen":

use case Vergehen erfassen
actors
Sachbearbeiter
precondition
Die Person, die das zu erfassende Vergehen begangen hat, ist bereits ausgewählt. Deren Daten sind überprüft und aktuell
main flow
Der Sachbearbeiter wählt zunächst das begangene Delikt aus (include "Delikt auswaehlen"). Nach erfolgter Auswahl gibt er im Bearbeitungsfenster die Daten für das zu speichernde Vergehen ein. Im Bearbeitungsfenster sollen aus Gründen der Anschaulichkeit auch relevante Attribute der betreffenden Person und des ausgewählten Delikts angezeigt werden. Nach Abschluss des Eingabevorgangs wird das Vergehen im System registriert.
Jetzt findet die Prüfung gemäß den Vorgaben statt. Das Ergebnis ist negativ, ein Entzug des Führerscheins ist also nicht erforderlich.
alternative flow
Bis einschließlich der Prüfung gleicher Verlauf wie beim main flow. Das Ergebnis der Prüfung ist positiv, daher wird jetzt der Anwendungsfall "Führerscheinentzug erfassen" gestartet, bei dem die Details des Entzugs festgelegt werden (extension point "Führerschein entziehen"). Das Erfassen des Entzugs verläuft erfolgreich.
postcondition
Das Vergehen ist erfasst, die Daten des Verkehrssünders sind aktuell, sofern erforderlich wurde der Führerschein entzogen.
exceptional flow
Gleicher Verlauf wie beim alternative flow, nur dass das Erfassen des Entzugs nicht erfolgreich ist.
postcondition
Das Vergehen ist erfasst, die Daten des Verkehrssünders sind aktuell, der erforderliche Entzug des Führerscheins wurde noch nicht durchgeführt.
end Vergehen erfassen

use case Führerscheinentzug erfassen
actors
Sachbearbeiter
precondition
Person ausgewählt, Führerscheinentzug erforderlich
main flow
Das Erfassen des Führerscheinentzugs findet gemäß den Vorgaben statt, der Vorgang ist erfolgreich.
postcondition
Lappen ist weg
exceptional flow
Das Erfassen des Führerscheinentzugs ist nicht erfolgreich, der Sachbearbeiter wird über einen Warnhinweis darauf aufmerksam gemacht.
postcondition
Der Führerscheinentzug ist nicht erfasst, der Sachbearbeiter wurde darauf hingewiesen.
end Sperre erfassen

extend relationship
base
"Vergehen erfassen"
extension point
Führerschein entziehen
extension
"Führerscheinentzug erfassen"
condition
Die an die Erfassung eines neuen Vergehens anschließende Prüfung hat ergeben, dass ein Entzug des Führerscheins erforderlich ist.
end

Anmerkung:
Kein Punktabzug, wenn der Misserfolgsfall von "Führerscheinentzug erfassen" nicht betrachtet wurde.

d) Erweitertes Sequenzdiagramm:

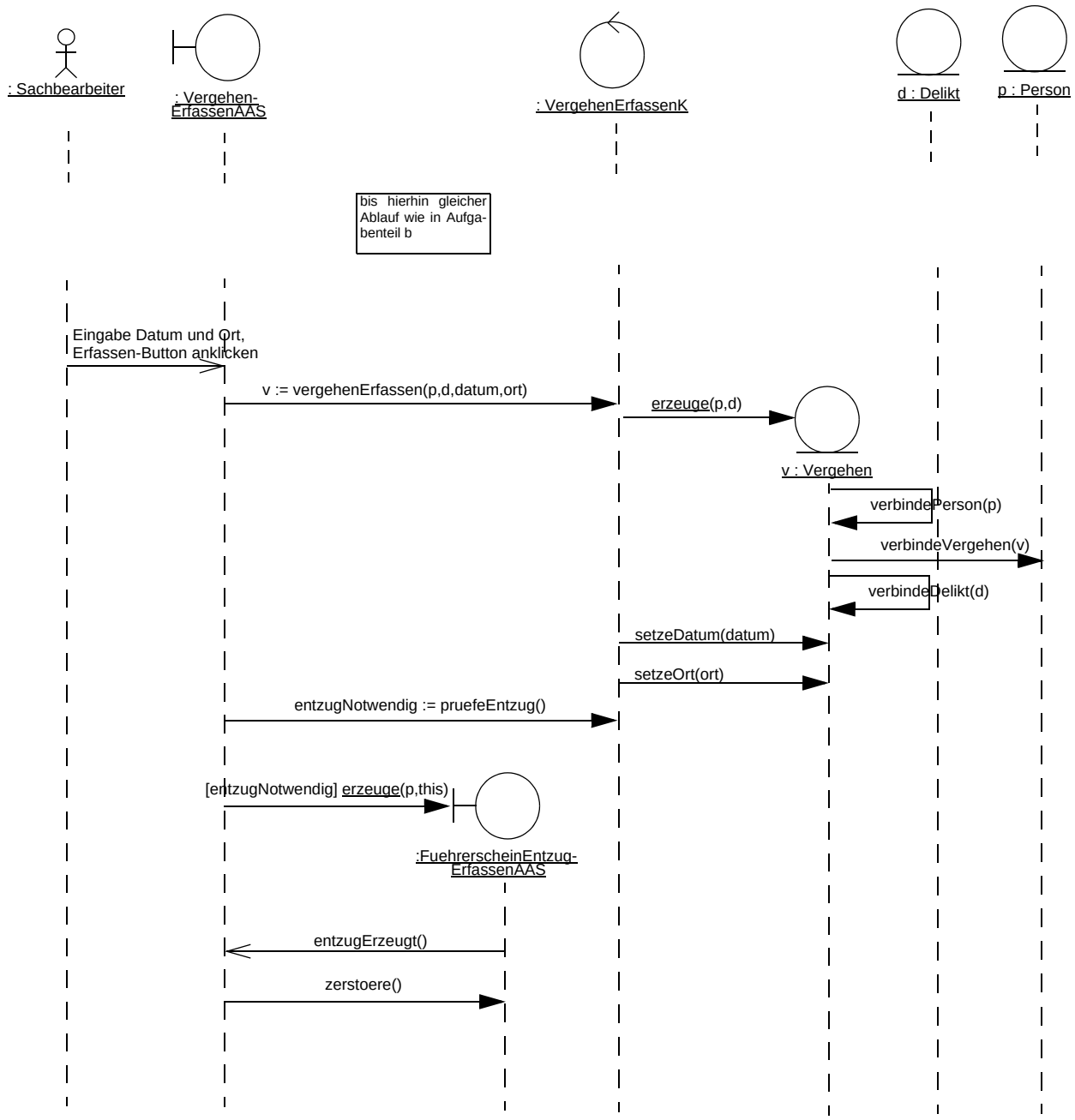

Abb. 4 erweitertes Sequenzdiagramm für das Szenario “Vergehen erfassen - erfolgreich“

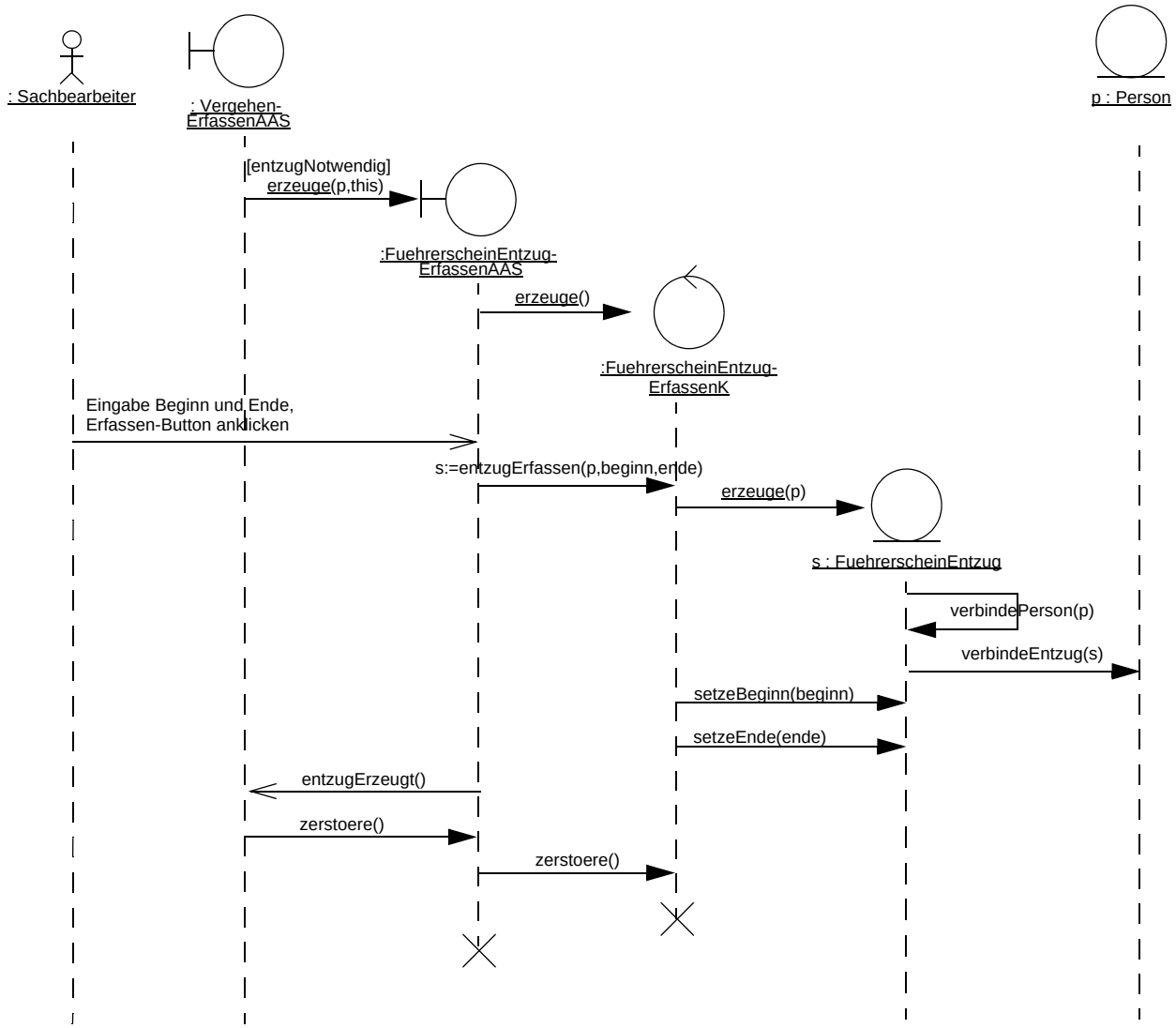

Abb. 5 Sequenzdiagramm für das Szenario §"Fuehrescheinentzug Erfassen - erfolgreich“

sinnvolle Alternativlösungen sind z.B.:

- Operation pruefeEntzug in der Klasse Person. Vorteil: vermeidet Redundanz, falls andere Anwendungsfälle die Prüfung der Sperre ebenfalls benötigen. Nachteil: Gehört eigentlich zur Programmlogik und damit nicht in eine Entitätsklasse.
- Operation vergehenErfassen ruft pruefeEntzug auf. Das FuehrerscheinEntzugErfassenAAS-Objekt wird dann von der VergehenK-Instanz erzeugt. Vorteil: Die Programmlogik befindet sich vollständig in der Kontrollklasse (In der ML ist das nicht der Fall, da pruefeSperre von der AAS-Instanz aufgerufen wird). Nachteil: Kontrollklassenobjekt erzeugt AAS-Klassenobjekt. Das widerspricht dem Prinzip der externen Kontrolle (das allerdings erst beim Grobentwurf festgelegt wird, hier befinden wir uns noch in der Analyse).

Aufgabe 2

a) In der Aufgabenstellung war ein Domänen-Klassendiagramm gefordert. Abb. 6 und Abb. 7 zeigen mögliche Lösungen.

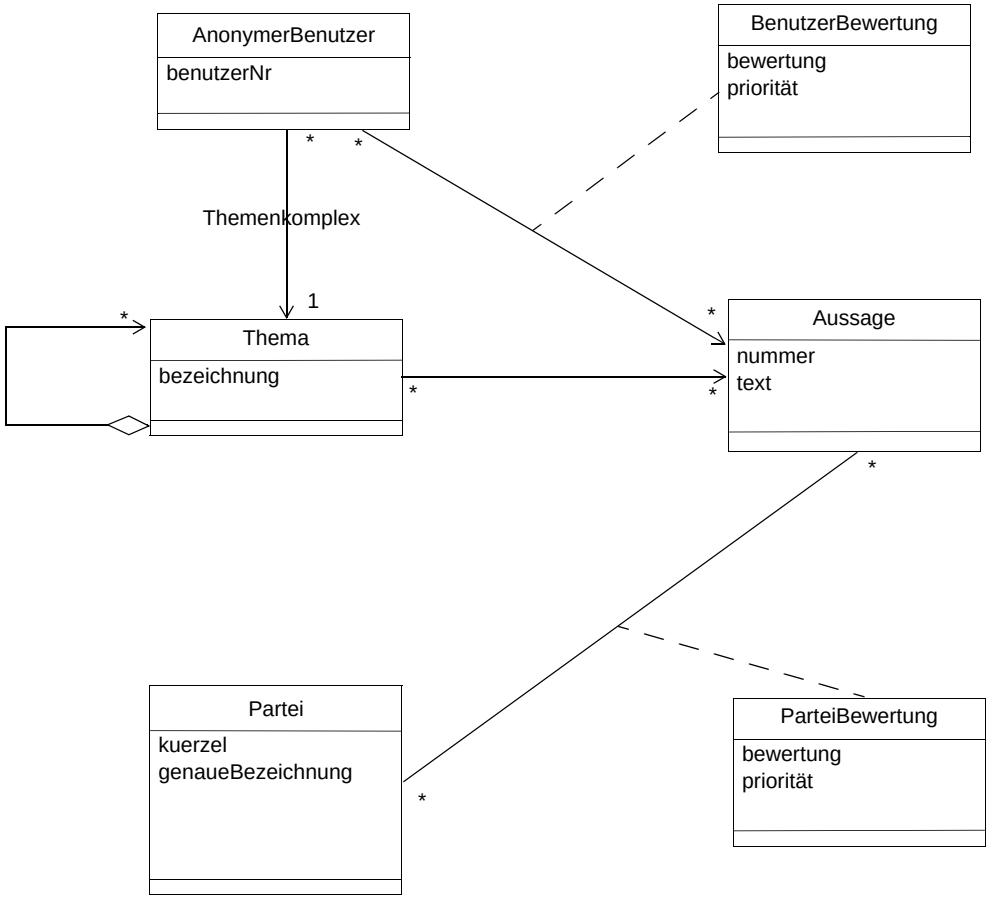

Abb. 6 Domänen-Klassendiagramm zur Aufgabe 2, Variante 1

Die Klasse Partei wird benötigt, da man mit möglichst geringem Änderungsaufwand Parteien anlegen oder entfernen können möchte. Diese Vorgänge sind so möglich, ohne dass an den anderen Klassen Anpassungen vorgenommen werden müssen. Bei einer Lösung beispielsweise mit "festen" Attributen in der Klasse Aussage wäre man in dieser Hinsicht wesentlich unflexibler.

Die einzelnen Aussagen werden von den Parteien und vom Benutzer gleichermaßen bewertet. Wir verwenden hier dafür getrennte Klassen, es sind jedoch auch andere Lösungen denkbar (s. Abb.6). Zur Redundanzvermeidung könnte man später eine abstrakte Klasse Bewertung als Superklasse von ParteiBewertung und BenutzerBewertung einführen.

Von der Klasse AnonymerBenutzer gehen nur unidirektionale Assoziationen aus, da in den assoziierten Klassen keine Informationen über den Benutzer benötigt werden und sie diesen daher nicht kennen müssen.

Es kann hier mit Assoziationsklassen gearbeitet werden, da zwischen je zwei Instanzen der verbundenen Klassen nie mehr als eine Verbindung besteht. Nach Vorgabe in der Aufgabenstellung darf keine Aussage dem Benutzer zweimal zur Bewertung angezeigt werden, daher ist dies auch für die BenutzerBewertung sichergestellt.

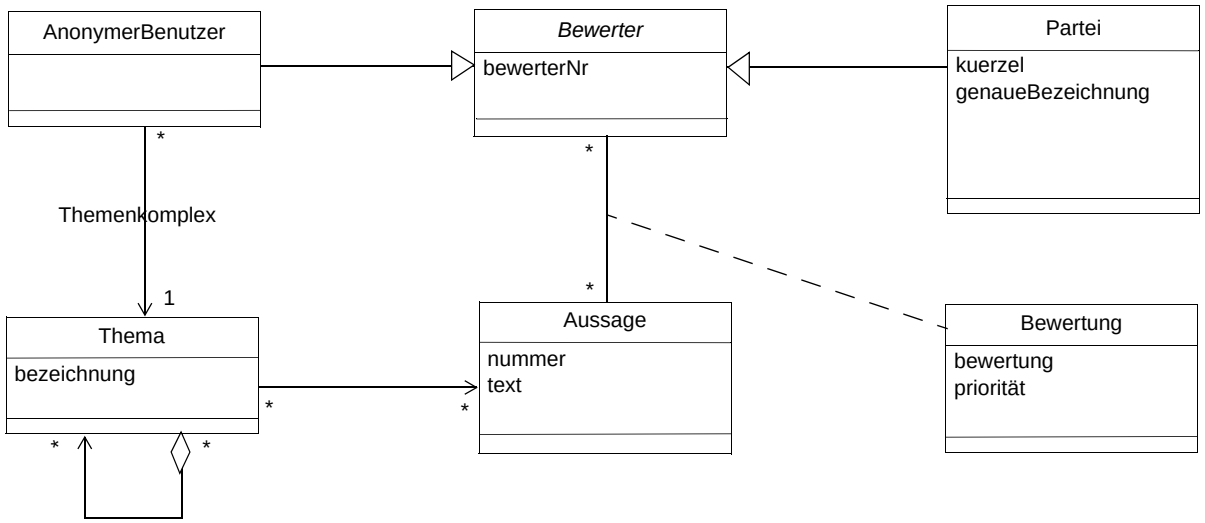

Abb. 7 Domänen-Klassendiagramm zur Aufgabe 2, Variante 2

b) Folgende Bedingungen in der Realität werden durch das Klassendiagramm in Abb. 6 nicht ausgedrückt:

- Die vom Benutzer zu bewertenden Aussagen gehören alle zum gewählten Thema.
- Eine Partei muss alle (zur Anzeige bereiten) Aussagen bewertet haben und umgekehrt.
- Ein Thema darf sich selbst nicht enthalten.
- Wenn Thema A Thema B enthält, darf Thema B nicht Thema A enthalten.

Aufgabe 3

a) zeigt ein Sequenzdiagramm, das dem Kollaborationsdiagramm in der Aufgabenstellung entspricht.

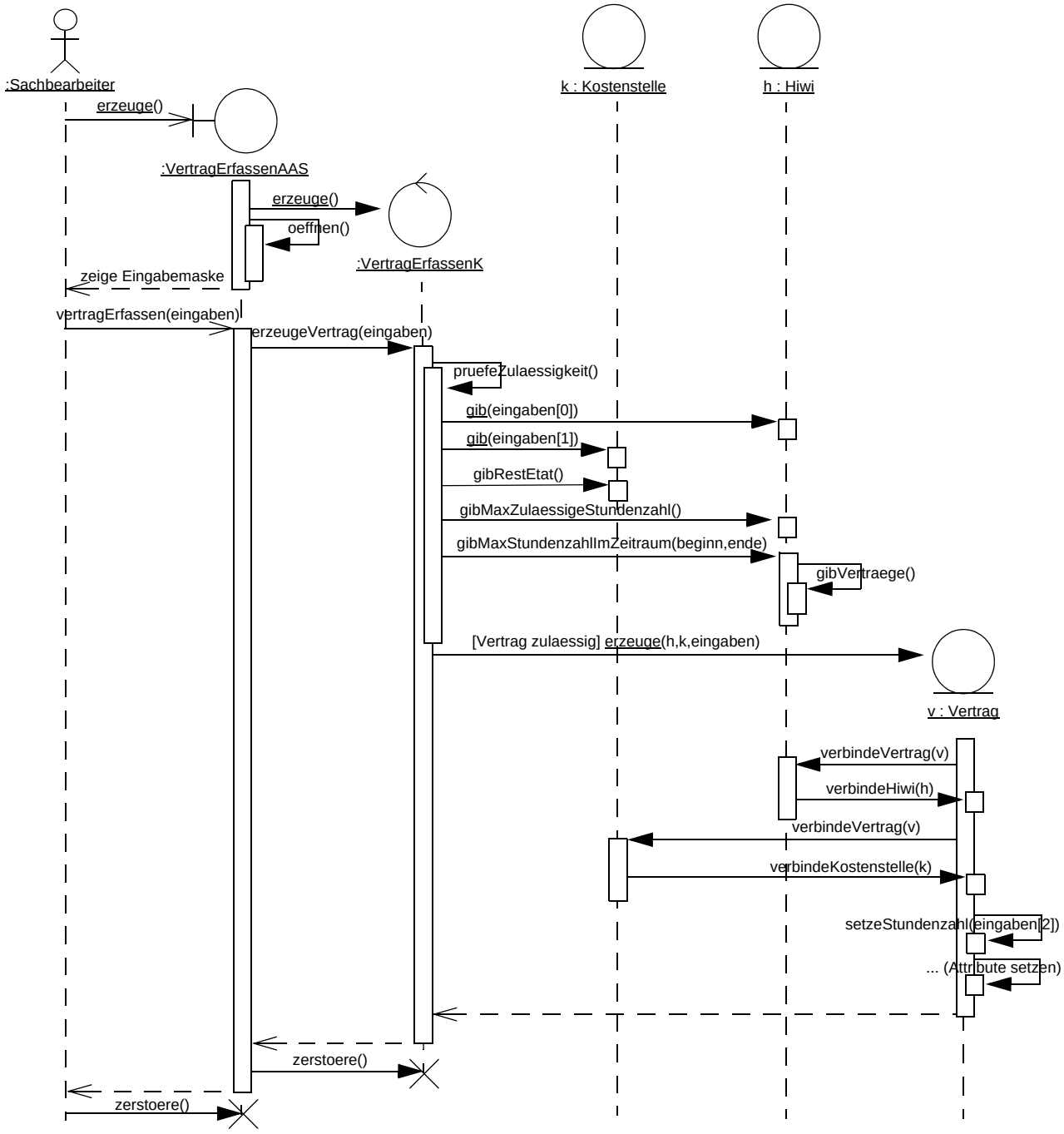

Abb. 8 Sequenzdiagramm zur Aufgabe 3a

b) Es wird nicht auf Kopien gearbeitet.

Aufgabe 4

a) Sowohl Benutzer als auch Rolle sind gemäß Kap. 48.3 des Kurstextes Echte Ganzes-Klassen, daher müssen die beiden Ordnerklassen BenutzerOrdner und RolleOrdner eingeführt werden. Die m:n-Assoziation wird nach den Empfehlungen in Kap. 48.1 mit Hilfe der Relationsklasse BenutzerRolleRelation und der Paarklasse BenutzerRollePaar transformiert.

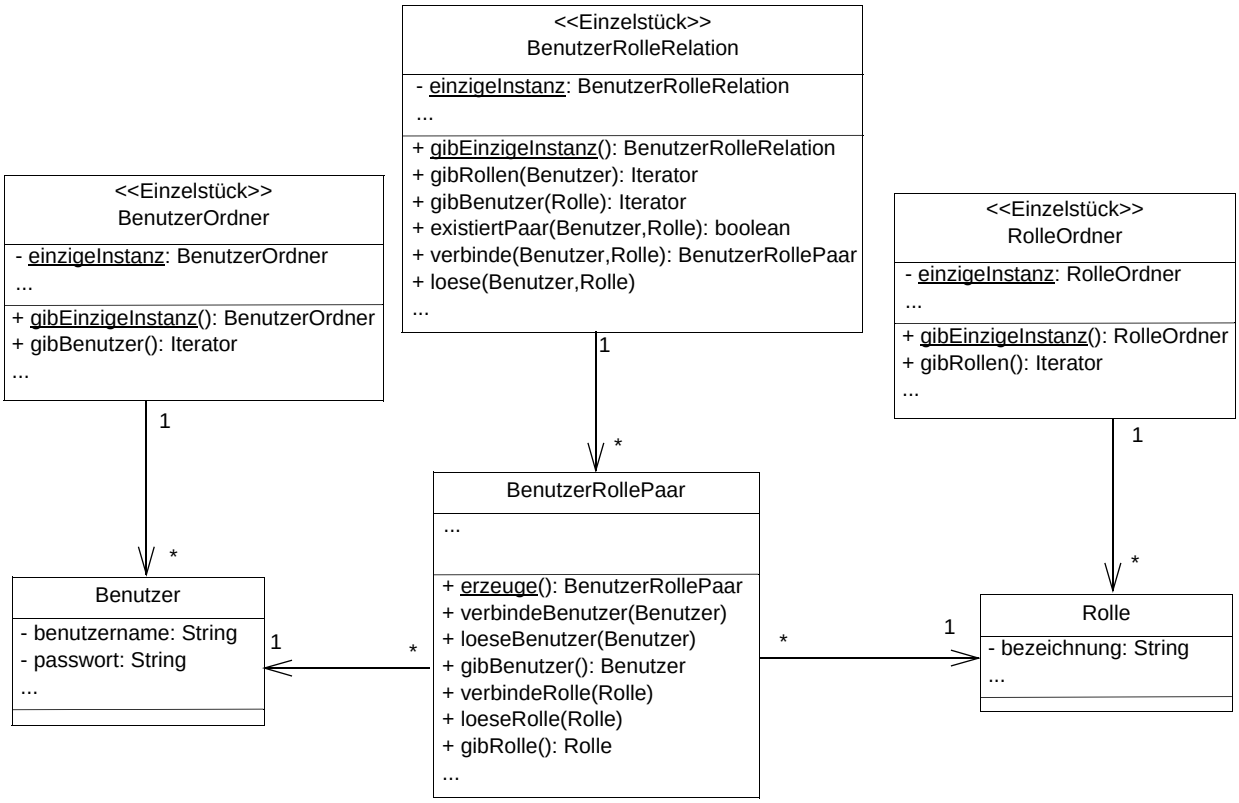

Abb. 9 Transformiertes Klassendiagramm zur Aufgabe 4a

b)

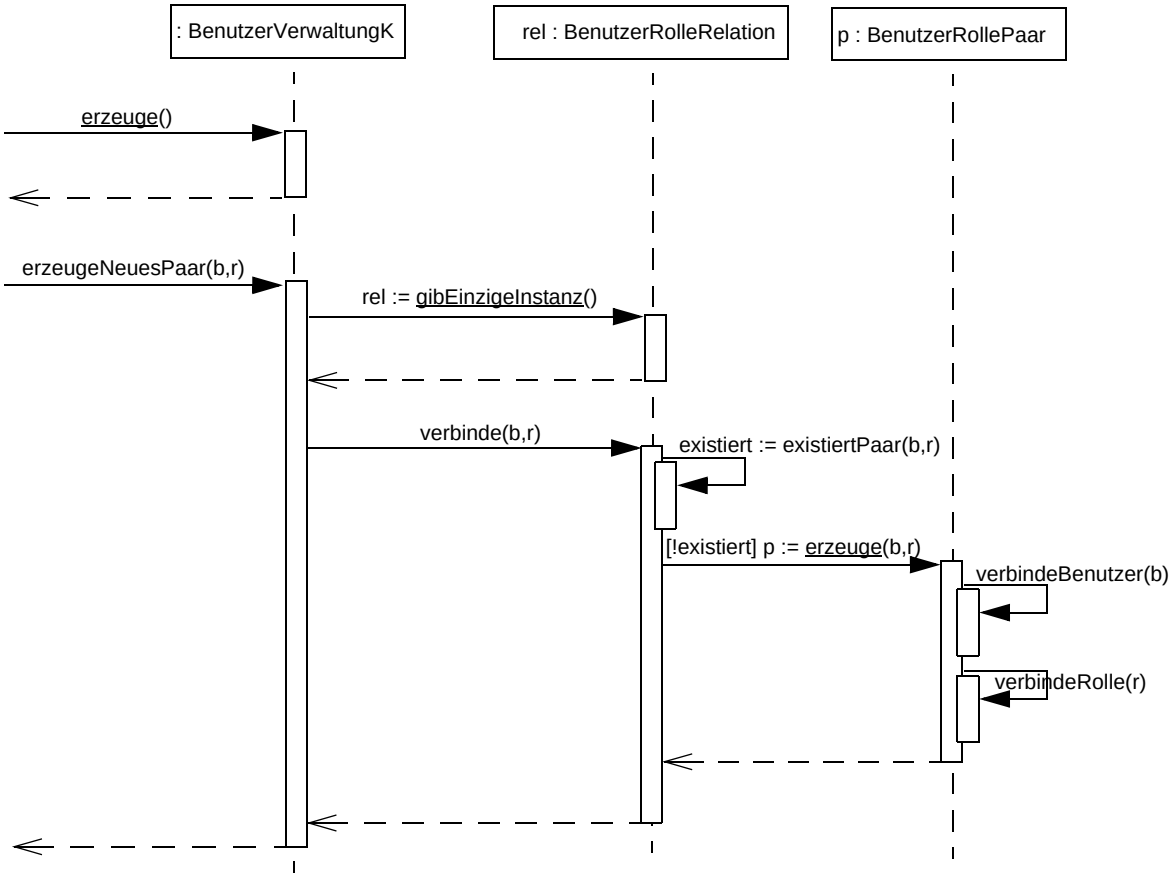

Abb. 10 Sequenzdiagramm zur Aufgabe 4b

c) Die Operation gibBenutzer(Rolle) in der Klasse BenutzerRolleRelation fällt weg.

d) Hier ist auch eine Transformation wie bei einer 1:n-Assoziation mit Navigierbarkeit in Richtung der Multiplizität * möglich.

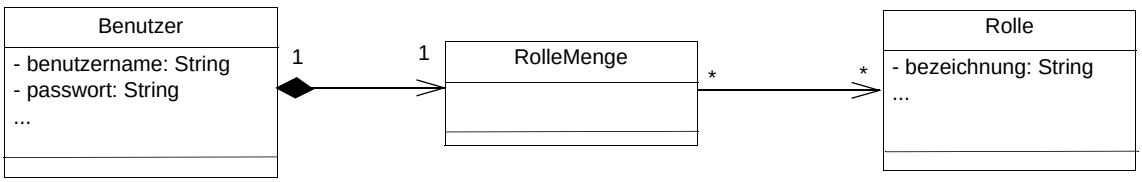

Abb. 11 Transformiertes Klassendiagramm zur Aufgabe 4d

Von einer Transformation mit Behälterklassen wurde bei bidirektionalen m:n-Assoziationen im Kurstext vor allem deswegen abgeraten, weil die Verantwortung für die Konsistenz der entsprechenden Verbindungen auf die Instanzen der assoziierten Ausgangsklassen (hier Benutzer und

Rolle) verteilt ist. Bei einer unidirektionalen Assoziation besteht dieser Nachteil nicht, daher ist die in Aufgabenteil d dargestellte Transformation naheliegender.

e) In diesem Fall haben wir eine unidirektionale Assoziation mit festem n. Bei einer geringen Anzahl an Rollen (hierüber wird in der Aufgabenstellung keine Aussage getroffen) kann beispielsweise für jede Rolle ein eigenes Attribut in der Klasse Benutzer vorgesehen werden. Bei einer größeren Anzahl an Rollen kann man z.B. ein Array vom Typ Rolle[] verwenden.

Kleine Anzahl an Rollen:

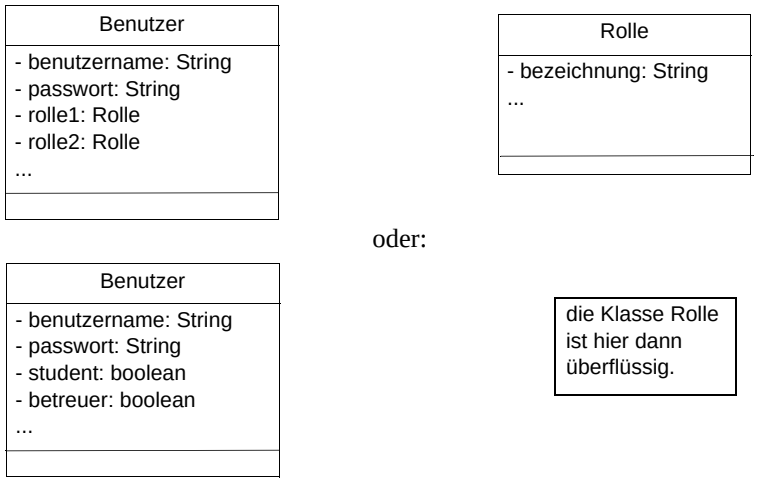

Abb. 12 Transformiertes Klassendiagramm zur Aufgabe 4e

Größere Anzahl an Rollen:

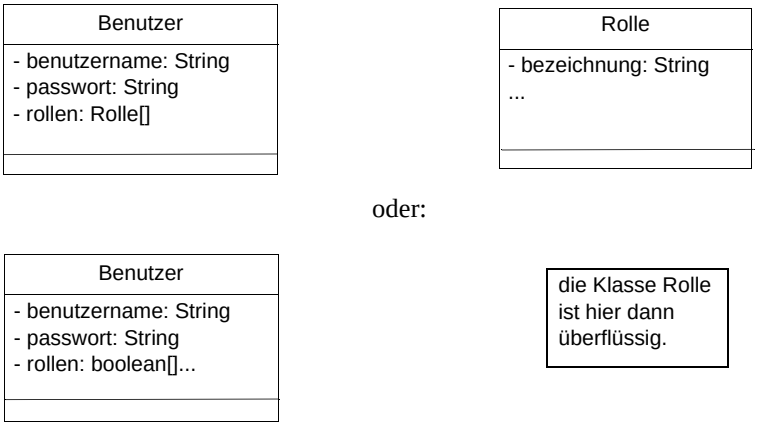

Abb. 13 Transformiertes Klassendiagramm zur Aufgabe 4e

In der Klausur gibt es die volle Punktzahl, wenn die Situation "festes n" erkannt und zwischen großem und kleinem n unterschieden wurde.