Robustness Analysis

Verwendeter Use Case:

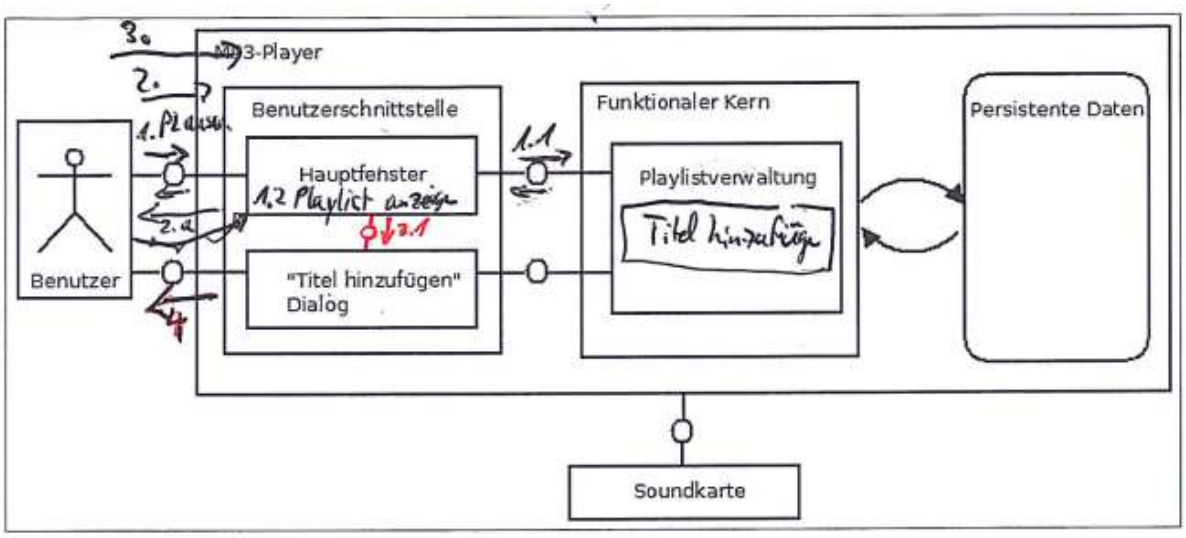
Interessant ist, dass beim Durchspielen in diesem Jahr mir aufgefallen ist, dass der ganze 1. Schritt eigentlich sehr ungenau ist und, dass hier einiges passiert (siehe v.a. auch Sequenzdiagramm).

Man sieht also, dass die Robustness Analysis echt auch sehr gut ist, um Unstimmigkeiten bei Anforderungen aufzudecken:

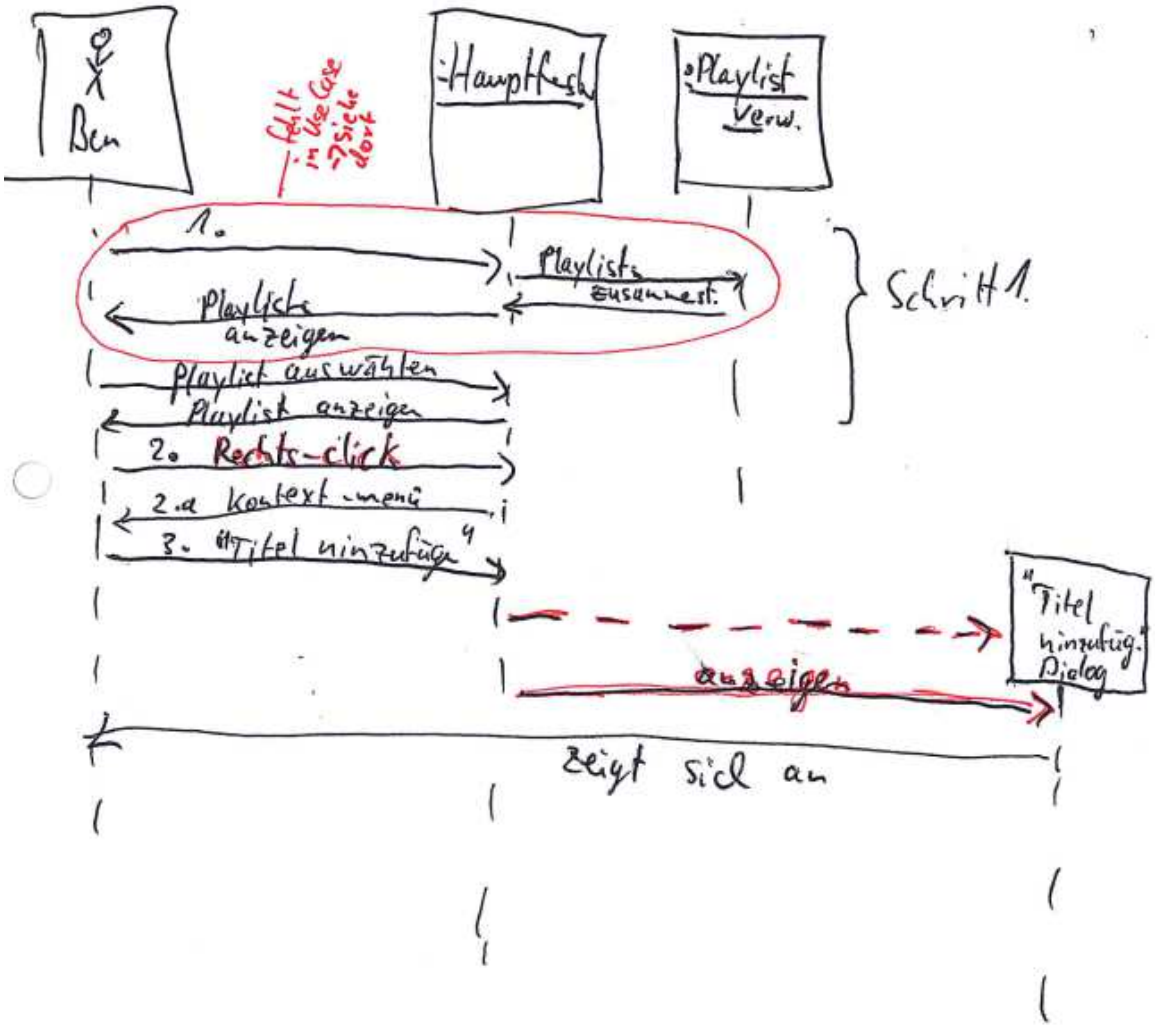
Use Case "Titel hinzufügen" (Standardablauf)

1. Der Anwender wählt im Hauptfenster eine Playlist aus.
2. Das System zeigt das Kontext-Menü an.
3. Der Anwender klickt auf "Titel hinzufügen" im Kontext-Menü.
4. Das System zeigt den Dialog "Titel hinzufügen" an.
5. ...

Kommunikationsdiagramm basierend auf dem verwendeten Grobdesign (Architektur):



Durchgespieltes Szenario als Sequenzdiagramm - wie in der Vorlesung:



Robustness Analysis WS 2015/2016

Die Robustness Analyse aus dem vorherigen Jahr. Diese ist noch ganz interessant wegen der Gedankengänge, die ich dann unten zu dem Auffinden der Unterschiede zw. Aufruf- und Signalsemantik geführt haben – lesen Sie das einmal aufmerksam durch.

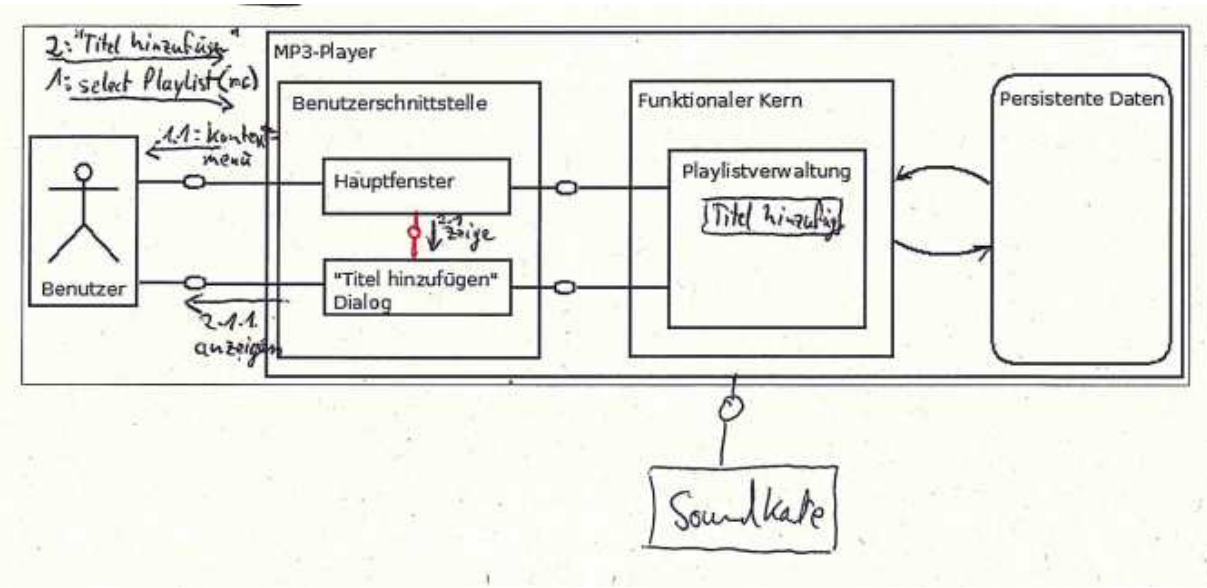
Verwendeter Use Case:

Use-Case "Titel hinzufügen" (Standardablauf)

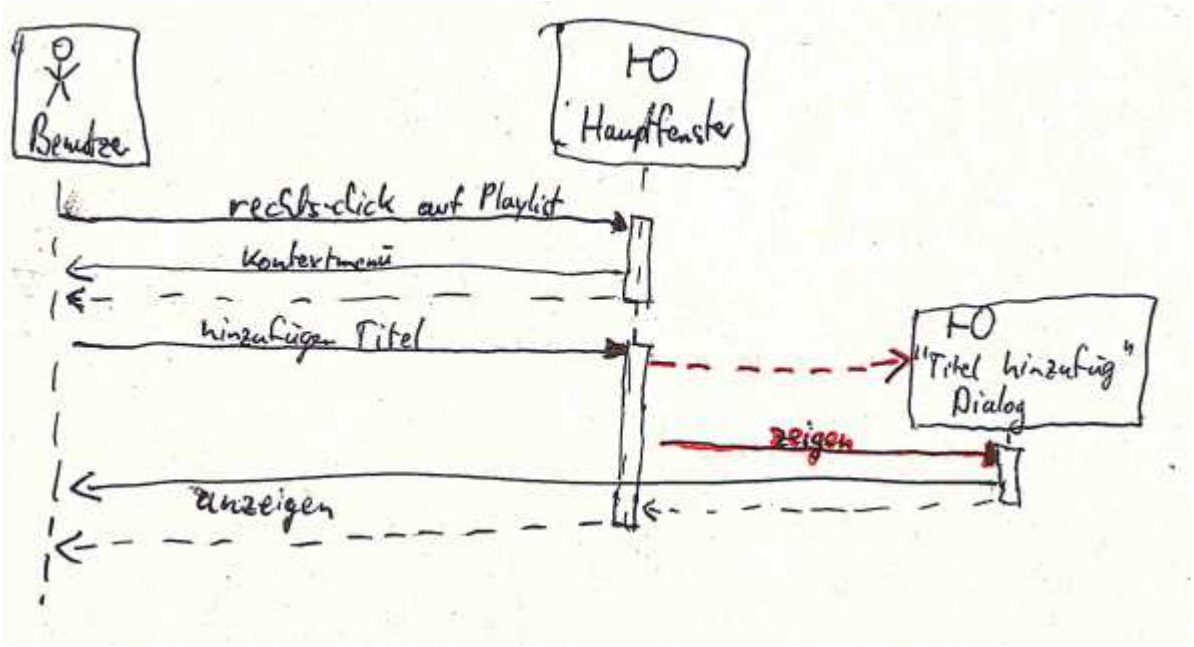
per rechts-click

1. Der Anwender wählt im Hauptfenster eine Playlist aus.
2. Das System zeigt das Kontext-Menü an.
3. Der Anwender klick auf "Titel hinzufügen" im Kontext-Menü.
4. Das System zeigt den Dialog "Titel hinzufügen" an.
5. ...

Kommunikationsdiagramm basierend auf dem verwendeten Grobdesign (Architektur):



Durchgespieltes Szenario als Sequenzdiagramm - wie in der Vorlesung:



Sequenzdiagramm1: Szenario in der Vorlesung unter der Verwendung der Aufruf-Semantik

Hinweise zu dem Szenario:

Ich habe versucht das Szenario nach UML 2.0 Vorgaben abzubilden. Die geschlossenen durchgezogenen Pfeile stellen entsprechend synchrone Aufrufe dar und gehen vom Benutzer aus.

Da GUI-Elemente normalerweise am Bildschirm angezeigt werden, jedoch dann nicht klar ist wann der Benutzer wirklich reagiert, sollte hier, wenn man es schon korrekt macht, ein asynchroner Pfeil (offener durchgezogener Pfeil) kommen.

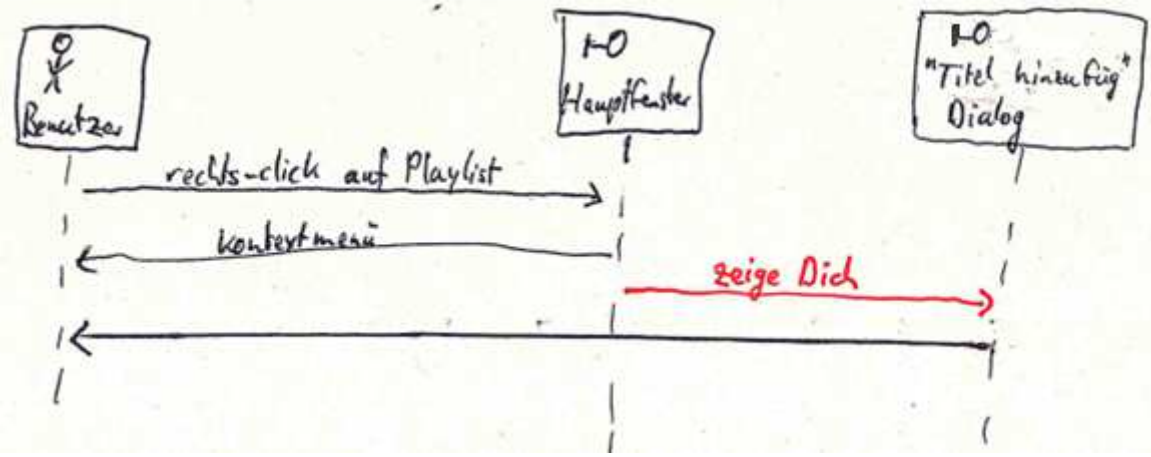
Nachteile:

- Der UML 2.x Standard **lässt leider nicht mehr zu** wie in UML 1.x die Rückgabepfeile bei synchroner Kommunikation **wegzulassen**, weshalb jetzt viele gestrichelte Pfeile nötig sind.
- Was mir jedoch noch weniger gefällt, ist die Tatsache, dass man sich ja noch in einer relativ frühen Phase befindet. Hier stellen die Festlegungen auf synchrone und asynchrone Aufrufe und die Erzeugung „Titel hinzufügen“-Dialogs tatsächlich die Gefahr einer Vorfestlegung dar, die ja erst dann im Detailed Design vielleicht erfolgen sollte. Das ist nicht optimal und tatsächlich kenne ich so aus der Praxis auch eher, dass man in diesen Phasen nur versucht das echt gesicherte Wissen über die Kommunikation festzulegen.
- Umgekehrt, jetzt einfach so die Pfeile für asynchrone Kommunikation zu verwenden (wie das in der Praxis so passiert) nur weil das einfacher ist, ist auch irgendwie unbefriedigend, weil man dann leicht eine Vorfestlegung auf asynchrone Kommunikation macht, die man nicht möchte.

Viel besser wäre hier tatsächlich, wenn man einen Pfeil hätte, der den Typ der Nachricht unbestimmt ließe. Entsprechend habe ich auch nochmals nachrecherchiert und, wenn man [UMLglasklar; S. 430]

genau liest, so gibt es neben den synchronen und asynchronen Methodenaufrufen tatsächlich einen weiteren Nachrichtentyp.
Dieser Typ bedeutet einfach, dass ein Signal zwischen zwei Kommunikationspartnern ausgetauscht wird. Leider ist dieser Typ so definiert, dass einfach angegeben ist, dass Signale asynchron sind und damit auch mit dem asynchronen Pfeil definiert werden [UMLglasklar; S. 430].
D.h., wenn man asynchrone Pfeile in den frühen Phasen verwendet, kann man sich hier darauf berufen, dass man hier zunächst nur über Signale spricht, hier aber noch unbestimmt ist, ob die Kommunikation synchron oder asynchron erfolgt. Mir persönlich würde hier ein eigener Pfeil, der anzeigt, dass noch unbestimmt ist, ob die Kommunikation asynch oder synch ist besser gefallen, aber den gibt es leider nicht.

Entsprechend mit der Signal-Semantik würde mir eigentlich in dieser frühen Phase (Anforderungs-phase und Architektur) doch Szenariodiagramm2 besser zusagen.



Sequenzdiagramm2: Szenario unter Verwendung der Signal-Semantik

Nochmals zusammengefasst:

Beide Sequenzdiagramme sind richtig.

Jedoch wäre Sequenzdiagramm2 vielleicht doch in den frühen Phasen mit dieser Signal-Semantik zu bevorzugen – man sollte hier jedoch dann beachten, dass in diesem Fall noch nicht festgelegt ist wie genau dieses Signal dann im System umgesetzt ist und damit auch nicht festgelegt ist, ob dies asychron oder synchron passiert!

Sequenzdiagramm1 wäre vielleicht doch eher dann ein Diagramm, das man dann in der Phase des Detaillierten Designs zeichnet, weil es dann auf die spezifischen technischen Details wie synchrone Aufrufe bei User-Klicks auf die GUI, jedoch asynchrone Anzeige der GUI beim Nutzer.