Software Technologie
Hausaufgabe

Pongrácz Vince
MKMDJO

2020

● Problembeschreibung

Ein Spiel muss man planen, das über Asteroidfordern geht. Ich schreibe hier auf, dass ich mit welchem anderen eigenen Beschränkungen gearbeitet habe.
In meinem Plan maximal 4 Spieler spielen gleichzeitig, und maximal 10 Roboter helfen die Astronauten.
Ein Astronaut (Spieler) kann Pause im Spiel haben, das dauert maximal 5 Minuten. Wenn alle Spieler Pause gedrückt haben, soll das Spiel nach 5 Minuten nach dem letzte Pauseanfrage beenden, wenn niemals kommt zurück.
In einem Asteroidloch können zwei Astronauten, oder Roboter sich gleichzeitig von dem Solarflair verstecken.
Ein Asteroidkern enthalt nur eine Einheit von einem Rohstoff, und ein Astronaut kann gleichzeitig nur eine Rohstoff transportieren.
Vor dem Solarflair soll das Spiel dieses Ereignis irgendwie signalisieren, um die Spieler sich verstecken zu können.
Fördern ist automatisch durchgeführt, wenn das Manteldicke beim Bohren 0 wird. In diesem Fall wird das Rohstoff im Container des Asteroids sein.

● Funktionale Anforderungen

2.1. Primäre Anforderungen

Id	**Beschreibung**	**Use-case-Name**	**Anmerkung**
R1	Die Spieler steuern die Siedler (die Astronauten)	bore asteroid transport resource fly to another asteroid shield	Siedler sind eigentlich die Astronauten
R2	Asteroid hat Mantel aus Stein	bore asteroid	Diese Mantel kann gebohrt werden
R3	Es gibt mehrere Rohstoffe	bore asteroid transport resource	(Rohstoffarten)
R4	Rohstoffe sind Wasser, Eis, Eisen, und radioaktives Material	bore asteroid transport resource	Noch andere Rohstoffe sind definierbar
R5	Ein Asteroid hat ein Kern	bore asteroid	
R6	Im Kern der Asteroiden kann man nur eine Rohstoff befinden, eine Einheit aus diesem Rohstoff	bore asteroid	Aber existiert leere Asteroid, dadrin ein Hohlraum befindet
R7	Die Asteroiden, die radioaktives Kern hat, sind gefährlich	bore asteroid	
R8	Es gibt solche Asteroiden, die keine Rohstoff im Kern haben, sondern Hohlraum	-	Die sind die leere Asteroiden
R9	Ein Asteroid hat nur eine Art von Rohstoff in sich selbst.	bore asteroid	Wenn es nicht leer ist
R10	In einem Schritt kann ein Astronaut (Spieler) nur eine Tätigkeit durchführen	-	

R11	Mögliche Tätigkeiten einer Astronaut sind: bewegen, bohren, fördern	fly to another asteroid bore asteroid transport resource	
R12	Beim Bewegen bewegt sich der Astronaut zwischen zwei Asteroiden	fly to another asteroid	
R13	Einer Asteroid kann gebohrt werden.	bore asteroid	
R14	Rohstoffe kann einer Astronaut zwischen zwei Asteroid transportieren (fördern)	transport resource	
R15	Jede Asteroid hat beliebig viele Nachbarn	-	
R16	Beim Bohren wird das Loch mit einer Einheit tiefer.	bore asteroid	Loch wird so modelliert, dass die Manteldicke kleiner wird
R17	Dicke der Mantel ist ein Eigenschaft der Asteroid	-	Manteldicke kann für unterschiedliche Asteroiden unterschiedlich sein
R18	Fördern ist dann und nur dann möglich, wenn das Loch fertig ist	bore asteroid	Loch ist fertig, wenn der Mantel ist völlig durchbohrt.
R19	Asteroid ist fähig zu explodieren.	bore asteroid	
R20	Asteroid hat Distanz von der Sonne gemessen	bore asteroid	
R21	Asteroid kann in der Sonnennähe, oder fern von der Sonne liegen.	bore asteroid	
R22	Wenn in der Sonnennähe liegende Asteroid radioaktive Rohstoff gebohrt wurde, Asteroid explodiert.	bore asteroid	Falls ein Astronaut bohrt, er ist gestorben
R23	Radioaktive Rohstoffe kann Astronaut nur von Sonnenfern liegende Asteroiden nehmen	bore asteroid	
R24	Solarflairs können die Asteroidzone erreichen, und die stellen eine Gefahr dar	-	
R25	Solarflair töten die Astronauten, und Roboter, wenn sie nicht im Hohlraum einer Asteroid sind	shield	
R26	Existieren Roboter, die helfen die Astronauten.	-	
R27	Roboter sind vom AI gesteuert.	-	Vom Spiel kontrolliert
R28	Mögliche Tätigkeiten einer Roboter sind: bewegen, bohren	fly to another asteroid bore asteroid	Roboter sind nicht fähig Rohstoff zu transportieren
R29	Roboter überleben die radioaktive Explosion, aber sie werden auf einer Nachbarasteroid geworfen.	bore asteroid	

R30	Das Spiel hat zwei Szenarien zum Spielende	-	Gewinnen, verloren
R31	Die Spielern haben verloren, wenn sie tot sind	-	wenn alle Spieler tot sind.
R32	Die Spieler haben gewonnen, wenn aus alle Rohstoffe mindestens eine Einheit gefördert, und diese Rohstoffe auf einer Asteroid transportiert wurde.	-	Weltraumbasis kann gebaut werden

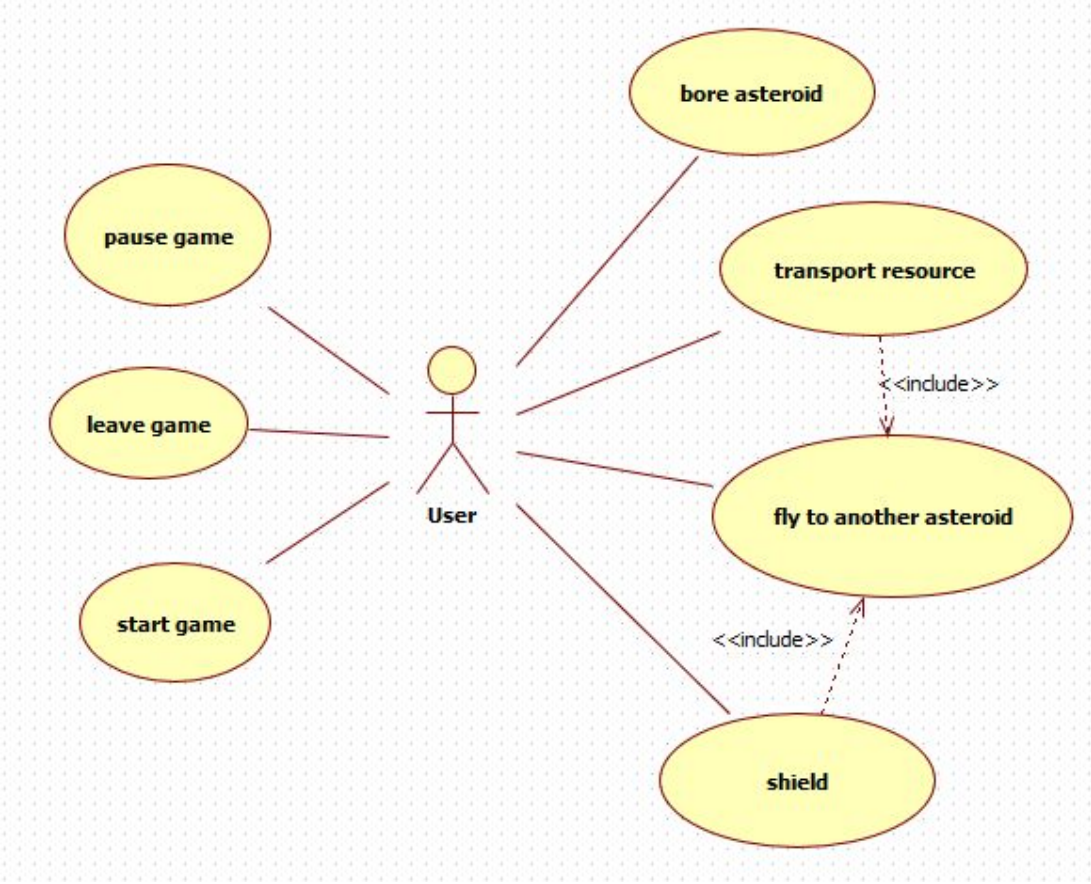
2.2. Zusätzliche Anforderungen

Id	**Beschreibung**	**Use-case-Name**	**Anmerkung**
R33	Für eine Astronaut gibt es Pausenfunktion im Spiel	pause game	Maximale Zeitintervall der Pause: 5 Minuten. Falls alle Spieler Pause haben, nach 5 Minuten der letzte Pauseanfrage soll das Spiel beenden.
R34	Maximal 4 Spieler sind gleichzeitig im Spiel erlaubt.	-	main, und Controller behandelt es
R35	Maximum 10 Roboter sind im Spiel erlaubt.	-	Controller behandelt es
R36	In einem Asteroidloch kann gleichzeitig nur 2 Astronaut, oder Roboter sein	shield	Als Asteroideigenshaft behandelt
R37	Solarflair ist ein Zufallsereignis	-	AsteroidZone.Sun behandelt
R38	Gleichzeitig kann ein Astronaut nur einen Rohstoff transportieren	transport resource	
R39	Wenn der Astronaut bewegt sich, er kann aus den Asteroiden wählen, dass sie auf welchem Asteroid fahren möchte.	fly to another asteroid	
R40	Wenn der Astronaut transportiert einer Rohstoff, er kann aus den Asteroiden wählen, dass sie auf welchem Asteroid Rohstoff transportieren möchte.	fly to another asteroid transport resource	
R41	Astronauten, und Roboter können sich in einer Hohlraum einer Asteroid verstecken.	shield	
R42	Ein Astronaut kann aus dem Spiel austreten	leave game	Diese Spieler ist als Tot betrachtet
R43	Die Spieler kommen nacheinander an der Reihe, diese wird als ein Schritt betrachtet	-	Reihenfolge ist zufällig
R44	Ein Asteroid kann gleichzeitig nur 2 Astronaut, oder Roboter in sich selbst	shield	

	verstecken		
R45	Beim Fördern wird das ganze Rohstoffmenge aufgefordert (eine Einheit aus dem Rohstoff).	bore asteroid	
R46	System soll die Astronauten über Solarflair vor dem Solarflair irgendwie berichten	-	

3. Anwendungsfall (Use-case) Modell

3.1 Use-case Diagramm



3.2 Beschreibung der Use-case

Use-case-Name (Titel)	Bore asteroid (Bohren eines Asteroids)
Kurze Beschreibung	Der Benutzer bohrt den Asteroid, auf dem er mit dem Astronaut steht.
Vorbedingungen	● allgemein: laufende Spiel ● zur Haupttätigkeit: Manteldicke der Asteroid ist nicht Null. ● zur A1 Alternative: Manteldicke ist Null. ● zur A2 Alternative: Manteldicke wird Null beim Bohren, und Asteroide hat radioaktive Material im Kern.
Akteur	User (Benutzer)
Haupttätigkeit	1. Der Benutzer drückt den BORE Taste. 2. Mantel der Asteroid wird mit einer Einheit kleiner. 3. System schreibt die aktuelle Manteldicke, und Zustandsinformationen der gebohrte Asteroid aus.
Alternative Tätigkeit	2. A1: System schreibt die aktuelle Manteldicke, und der Zustandsinformationen der Asteroide (hat es Rohstoff, oder nicht) aus. 3. A1 : Kernmaterial wird im Container des Asteroids 3. A2: Astronaut ist gestorben
Ausgangsbedingung	● zur Haupttätigkeit: Mit einer Einheit ist der Manteldicke kleiner. Neue Manteldicke, und Zustand des Asteroids ist ausgeschrieben. ● zur alternative Tätigkeiten: Manteldicke, und Zustand des Asteroids ist ausgeschrieben.

Use-case-Name (Titel)	Transport resource (Transportieren einen Rohstoff)
Kurze Beschreibung	Benutzer (Astronaut) transportiert eine Rohstoff von dem aktuellen Asteroid auf einem ausgewählten Asteroid.
Vorbedingungen	● allgemein: laufende Spiel ● zur A1: Im Container des aktuelles Asteroid keine Rohstoff befindet sich
Akteur	User (Benutzer)
Haupttätigkeit	1. Benutzer druckt den TRANSPORT Taste. 2. Benutzer wählt eine Material aus dem Container des aktuelles Asteroid. 3. Benutzer nutzt das Use-case von Fliegen nach einem anderen Asteroid. 4. Rohstoff wird auf dem gewählten Asteroid sein. (Rohstoff wird im Container des Zielasteroids erscheinen)
Alternative Tätigkeit	2. A1: Fehlermeldung wird ausgedrückt 3. A1: Benutzer muss andere Tätigkeit wählen.

Ausgangsbedingung	Astronaut ist auf dem Zielasteroid befindet. Rohstoff ist in dem Container des Zielasteroids. Die ausgewählte Rohstoff wird nicht dupliziert, aus dem Container des Herkunftsasteroids wird gelöscht.

Use-case-Name (Titel)	Fly to another asteroid (Fliegen nach einem anderen Asteroid)
Kurze Beschreibung	User wählt den Zielasteroid aus, und wird dorthin fliegen.
Vorbedingungen	allgemein: laufende Spiel
Akteur	User (Benutzer)
Haupttätigkeit	● Benutzer druckt das FLY Taste. ● Benutzer wählt eine beliebige Asteroid aus (in diesem Schritt sieht er manche Informationen über Asteroiden) ● Benutzer wird auf dem gewählte Asteroid fliegen
Alternative Tätigkeit	-
Ausgangsbedingung	Astronaut befindet sich auf einem anderen Asteroid (Zielasteroid)

Use-case-Name (Titel)	Go under shelter (In einem Asteroidhohlraum gegen Solarflair gehen)
Kurze Beschreibung	Wenn ein Solarflair kommt, Astronauten und Roboter auch sollen in einem Hohlraum einer Asteroid zu kehren, um nicht zu sterben
Vorbedingungen	allgemein: ● laufende Spiel. ● System hat ein Nachricht über Solarflair schon früher gesendet. ● Diese Nachricht ist sehbar bis Ende eines Solarflairs. zur A1: ● Der Asteroid kann nicht den Astronaut schützen.
Akteur	User (Benutzer)
Haupttätigkeit	1. Benutzer nutzt das Use-case Fliegen nach einem anderen Asteroid. 2. Benutzer wird automatisch geschützt (wenn der Asteroid darauf fähig ist)
Alternative Tätigkeit	2. A1: Benutzer wird über Schutzunfähigkeit benachrichtigt. 3. A1: Benutzer muss eine neue Asteroid finden (die erste 2 Schritte wiederholen). 2. A2 (Astronaut hat 3-mal keine Schutz gefunden) 3. A2 Astronaut ist gestorben.

Ausgangsbedingung	Astronaut ist geschützt, oder gestorben.

Use-case-Name (Titel)	Pause game
Kurze Beschreibung	Spieler kann ihre eigene Astronaut aus dem Spiel maximal 5 Minuten lang ausnehmen, als Pause.
Vorbedingungen	zur Haupttätigkeit: ● Spiel ist laufend, also es ist schon gestartet. zur Alternative A1: ● Spiel ist laufend, und der Astronaut des Benutzers ist schon IN_PAUSE. zur Alternative A2: ● Spieler kommt nicht zurück im Spiel nach 5 Minuten.
Akteur	User (Benutzer)
Haupttätigkeit	● Benutzer druckt PAUSE Taste. ● Spiel für diese Spieler ist gestolpert (Zustand der Astronaut wird IN_PAUSE)
Alternative Tätigkeit	1. A1 Benutzer druckt PAUSE Taste. 2. A1 Spiel wird für diese Benutzer weiterlaufen (Zustand der Astronaut wird DEFAULT) 2.A2: Spieler ist gestorben.
Ausgangsbedingung	-

Use-case-Name (Titel)	Leave game (Spiel weglassen)
Kurze Beschreibung	Eine Spieler tritt aus dem Spiel.
Vorbedingungen	Spiel ist gelaufen. (Spieler ist im Spiel)
Akteur	User
Haupttätigkeit	1. Spieler druckt das LEAVE Taste. 2. Astronaut wird gestorben. 3. Spieler lasst das Spiel weg.
Alternative Tätigkeit	-
Ausgangsbedingung	Spieler spielt schon nicht.

Use-case-Name (Titel)	Start game (Spiel anfangen)
Kurze Beschreibung	Benutzer signalisiert, dass er fertig zum Spielstart. Er wartet bis Spielanfang. Spielstart soll nach maximal 30 Sekunden später geschehen. In diesem kurzer Zeitintervall können andere Spielern sich im Spielvorraum eintreten.
Vorbedingungen	Programm ist gelaufen.
Akteur	User
Haupttätigkeit	1. Benutzer druck START GAME Taste. 2. Spieler wartet 3. 3. Spiel fangt an
Alternative Tätigkeit	3. A1 Keine weitere Spieler kommen in 30 Sekunden 4. A1 Spieler spielt nur mit Roboter. 3. A2 Keine weitere Spieler kommen in 30 Sekunden 4. A2 Spieler tritt aus, er möchte nicht nur mit Roboter spielen.
Ausgangsbedingung	-

● Strukturbeschreibung

●.1 Klassenbeschreibungen

4.1.1. Asteroid

- ***Zuständigkeit***

Repräsentiert ein Asteroid im Spiel. Diese Klasse kann die Spieler schützen, und verschiedene Rohstoffe speichern. Asteroid ist fähig zu explodieren, und Informationen über sich selbst liefern.

- ***Attributen***

+layer	Repräsentiert den Mantel des Asteroids
-deployedResources	Kollektion über die hier transportierte, und gebohrte Rohstoffe.
+core	Repräsentiert den Kern des Asteroids
-distanceFromSun	Distanz von der Sonne gemessen
-shieldCounter	Anzahl der aktuell geschützte Spieler
-shieldCounterMaximum	Maximale Anzahl der schützbare Spieler

- ***Methoden***

+explode()	Zerstörung der Asteroid (und dadurch die Membervariablen)
+deployResource(resource: ResourceType)	Liegt einen Rohstoff im Container des Asteroids
+listDeployedResources()	Leistet Informationen über Rohstoffe z.B: für GameEndObserver, oder für getInfo()
+hide():boolean	Repräsentiert die Schutzfunktion des Asteroids. Richtig, wenn der AbstractActor geschützt ist, falsch, wenn nicht.
+getInfo()	Leistet Informationen über das ganze Asteroid
+removeFormResources(resource :ResourceType)	Entfernt einen Rohstoff vom Container.

4.1.2. Layer

- ***Zuständigkeit***

Repräsentiert der Mantel eines Asteroids. Beim Bohren hat diese Klasse wichtige Rolle.

- ***Attributen***

-layerThickness:Integer	Manteldicke
-layerMaterial:String	Typ der Mantel (Grundeinstellung ist Stein)

- ***Methoden***

+thinIt():Integer	Verkleinert der Manteldicke, und gibt die aktuelle Manteldicke zurück

4.1.3. Core

- ***Zuständigkeit***

Repräsentiert des Kernes eines Asteroids. Diese Klasse speichert das Kernmaterial, und diese Klasse hat auch Rolle beim Bohren.

- ***Attributen***

-resource:ResourceType	Im Kern liegende Rohstoff

- ***Methoden***

+getResource():ResourceType	Gibt den Rohstoff an, aber fordert nicht den aus.
+boreOut():ResourceType	Nimmt den Rohstoff aus dem Kern aus, falls es möglich ist.

4.1.3. ResourceType

- ***Zuständigkeit***

Enumeration für verschiedene Kernmaterialien, bzw. Rohstoffen
Mögliche Rohstoffe:

- WATER
- ICE
- IRON
- RADIOACTIVE

- EMPTY

Anmerkung: EMPTY symbolisiert den Hohlraum

4.1.4. AbstractActor

- ***Zuständigkeit***

Abstrakte Basisklasse für gleiche Eigenschaften von Roboter, und Astronauten.

Schreibt vor, oder implementiert die Grundfunktionen von Roboter, und Astronauten.

Implementiert das Observer Interface (Schnittstelle).

- ***Attributen***

-state:AbstractActorState	Zustand der AbstractActor
-onAsteroid:Asteroid	Asteroid, auf dem der Spieler sich jetzt befindet.

- ***Methoden***

-chooseDestinationAsteroid(): Asteroid	AbstractActor wählt eine neue Asteroid, und sieht alle Informationen über Asteroiden
-reactExplosion()	Reaktion auf einer Explosion. Bei den Astronauten: Tod, bei den Roboter: auf einen anderen Asteroid weggeworfen
+fly()	Fliegen nach einem anderen Asteroid
+bore()	Bohren den aktuellen Asteroid (auf dem der AbstractActor steht)
+goForShield()	In einem Hohlraum sich verstecken

4.1.5. Astronaut

- ***Zuständigkeit***

Beschreibt das Verhalten einer Spieler (Astronaut). Eine Realisation/vererbte Klasse der AbstractActor Klasse.

- ***Methoden***

-chooseResourceType	Wählt eine Rohstoffart von dem Container des aktuellen Asteroids aus.
+transport	Transportiert eine Rohstoff von einem Asteroid zu eine andere Asteroid
+leave	Weglassen das aktuelle Spiel
+pause()	Pause bitten, maximal 5 Minute.

4.1.6. Observer

- ***Zuständigkeit***

Interface für Observer. Zum Überleben eines Solarflairs ist es nötig, für Solarflairrealisation muss man diese Interface verwenden.

- ***Methoden***

+notifyFlair()	Benachrichtigen die Observern über Solarflairereignis

+notifyDanger()	Benachrichtigen die Observern über kommende Solarflair

4.1.7. AIRobot

- ***Zuständigkeit***

AI (oder vom Spiel) gesteuerte Roboter, die die Astronauten helfen. Vererbte Klasse aus AbstractActor, wisst nichts mehr als AbstractActor.

- ***Methoden***

  Dieselbe wie beim AbstractActor, aber anders implementiert.

4.1.8. AbstractActorState

- ***Zuständigkeit***

Enumeration für verschiedene AbstractActionzustände
Mögliche Rohstoffe:

- PROTECTED → Dieser Zustand wird aufgenommen, wenn der Abstractactor schon von der Solarflair geschützt ist
- IN_DANGER → Dieser Zustand ist vor dem Solarflair aktiviert
- DEFAULT → Grundzustand, wenn keine Solarflair in der Nähe ist
- IN_PAUSE → Zustand, wenn ein Spieler im Pause ist
- DEAD → Signalisiert, wenn ein Spieler/Roboter gestorben ist. Für Austreten ist es wichtig. Solche AbstractActoren werden gelöscht, wo der Zustand DEAD ist.

4.1.9. Controller

- ***Zuständigkeit***

  Kontrolleinheit, diese beinhaltet die Astronauten, und die Roboter. Eine Singleton Klasse, also nur eine aus diese kann gleichzeitig existieren.

  Initialisiert das Spiel, und die tote Spieler wirft es weg.

- ***Attributen***

+astronauts:Astronaut[0..4]	Kollektion aus Astronauten, also die Spielern, oder Karaktern
+endObserver: GameEndObserver	Überprüft immer die Spielendebedingungen, und die Toten
+bots:AIRobot[0..10]	Kollektion aus Roboter, die im Spiel die Spieler helfen.
-instance:Controller	Instancereferenz wegen Singletonverhalten (static)

- ***Methoden***

+getInstance:Controller	Gibt zurück das aktuelle Controller
+removeAIActor(iactor:Abstract Actor)	Löscht eine AbstractActor, falls ihre Zustand DEAD ist.
+initGame()	Initialisiert das Spiel.

4.1.10. GameEndObserver

- ***Zuständigkeit***

  Prüft, und beobachtet das Spielende, also wenn jede Spieler gestorben ist, oder wenn die genügende Rohstoffmenge auf einem Asteroid zur Verfügung für Spielende steht.

- ***Methoden***

+resourceCheck()	Prüft, ob die genügende Rohstoffmenge auf einem Asteroid zur Verfügung für Spielende steht.
+deadCheck()	Beobachtet die Spielerzustände, und ruft removeIActor, wenn ein Spieler tot ist.

4.1.11. AsteroidZone

- ***Zuständigkeit***

  Kontrolleinheit, diese beinhaltet die Asteroiden, also diese ist das Asteroidenfeld. Eine Singleton Klasse, also nur eine aus diese kann gleichzeitig existieren.

  Gibt Information über sich selbst, also von alle Asteroiden, bestimmt das Sonnennähe, gibt eine nächste Asteroide zurück (als Nachbarasteroide), und kann eine neue Asteroid zum Asteroidenfeld addieren.

- ***Attributen***

+asteroids:Asteroid[0..*]	Kollektion aus Asteriden, die das Asteroidenzone bilden.
-distanceBound: Integer	Sonnennäheparameter. Wenn etwas naeher als diese Wert ist, dann es ist in der Sonnennähe.
+sun:Sun	AsteroidZone beinhaltet auch die Sonne, weil es ist Teil des Systems
-instance: AsteroidZone	Instancereferenz wegen Singletonverhalten (static)

- ***Methoden***

+getInstance:Controller	Gibt zurück das aktuelle Controller
+removeAIActor(iactor:Abstract Actor)	Löscht eine AbstractActor, falls ihre Zustand DEAD ist.
chooseNectAsteroid(): Asteroid	Gibt das nächste Asteroide zurück in dem Asteroidenzone.
+getZoneInfo()	Gibt Informationen über Asteroiden, und Asteroidenzone

4.1.12. Observable

- ***Zuständigkeit***

Interface für Observable. Wegen Observer, Observable Modell. Zum Solarflair wird es benutzt.

- ***Methoden***

+register(observer: Observable)	Registriert eine Observer auf dem Observable (wahrscheinlich addiert sie zu einer Kollektion)
+unregister(observer: Observable)	Löscht der Registration einer Observer.

4.1.13. Sun

- ***Zuständigkeit***

Repräsentiert Sonne, implementiert Observable, um Solarflair zu signalisieren, und benachrichtigen.

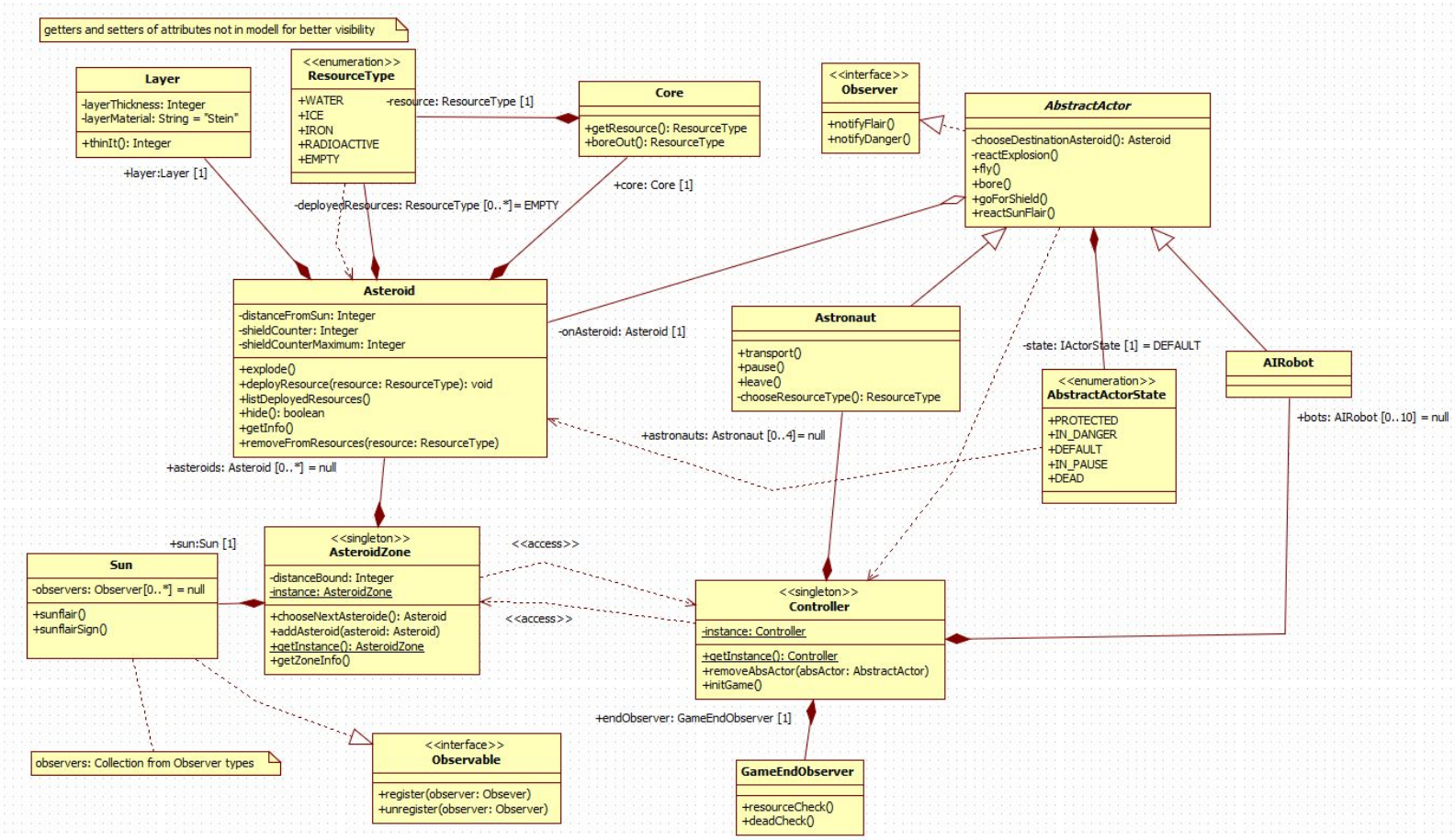
- ***Attributen***

-observers: Observer[0..*] = null	Kollektion aus Observers. (alle, die implementieren Observer, hier also AbstractActoren). Am Anfang auf Null gesetzt, aber alle Observer können registrieren.

- ***Methoden***

+sunflair()	Durchführt das Solarflair
+sunflairSign()	Signalisiert eine Solarflair, dass es kommen wird.

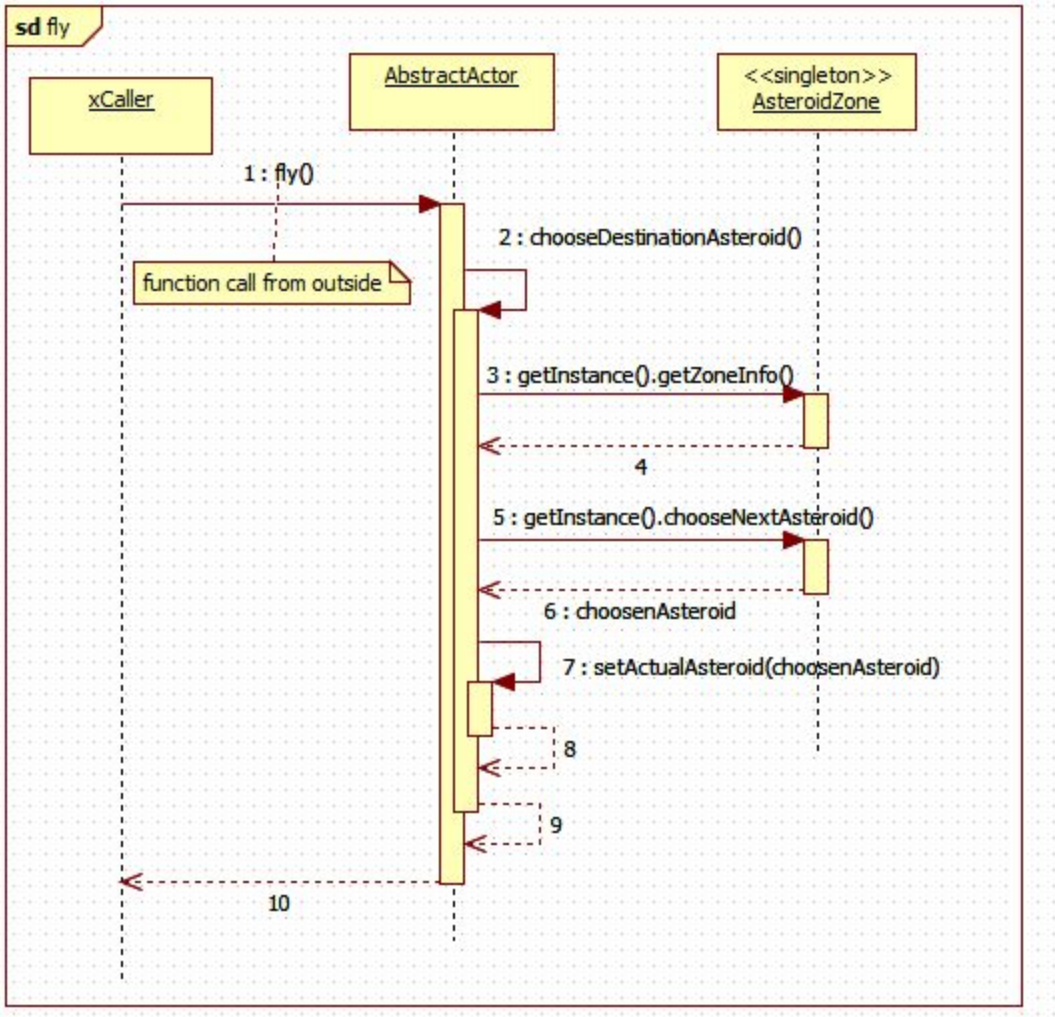
●.1 Klassendiagramm



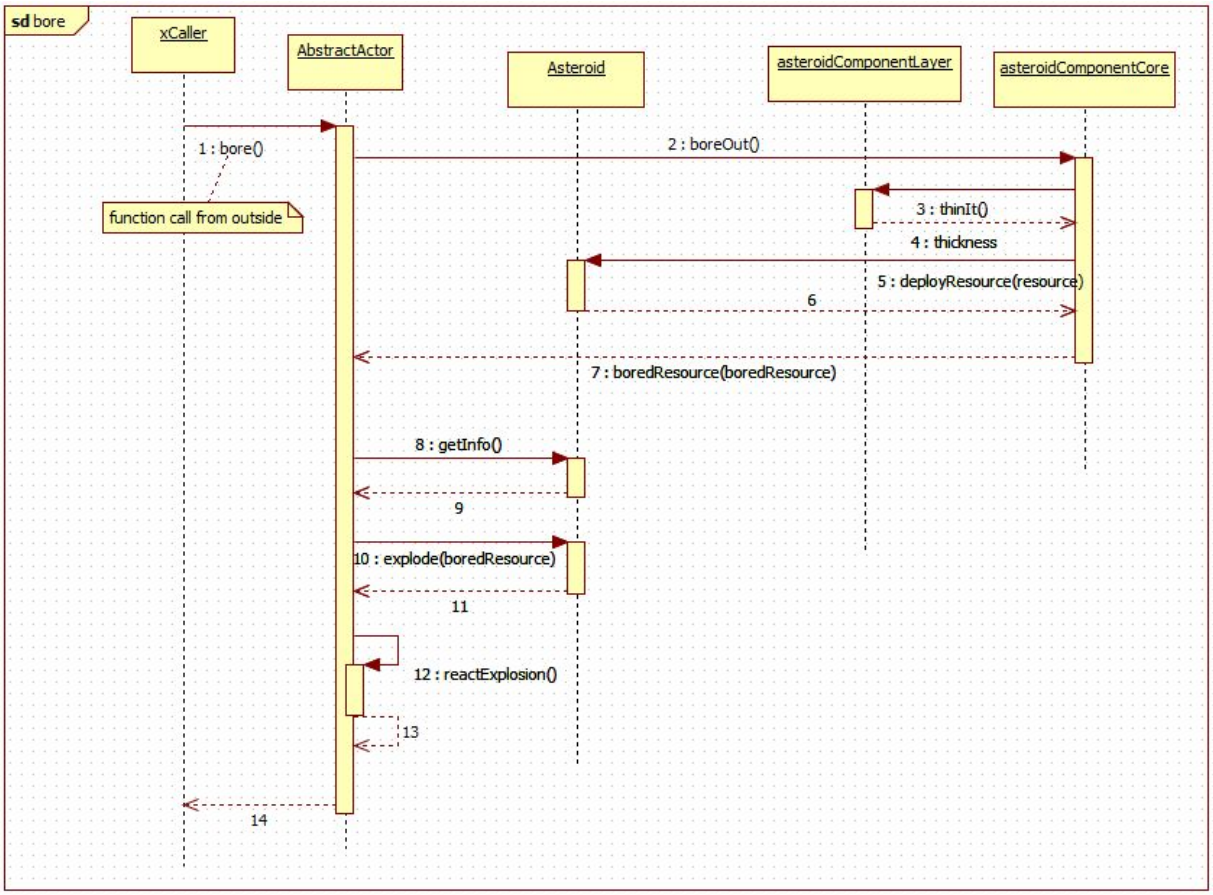
● Funktionale Beschreibung

●.1 Ablaufdiagramme

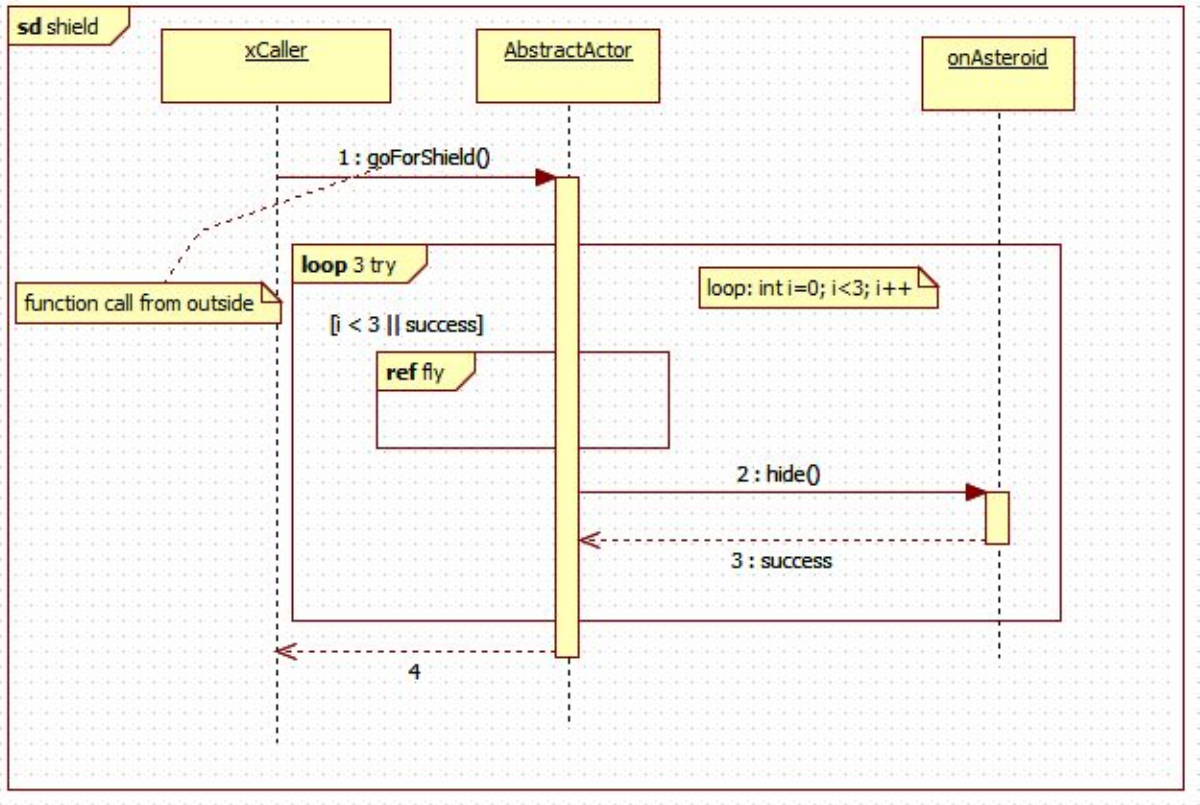
5.1.1. fly



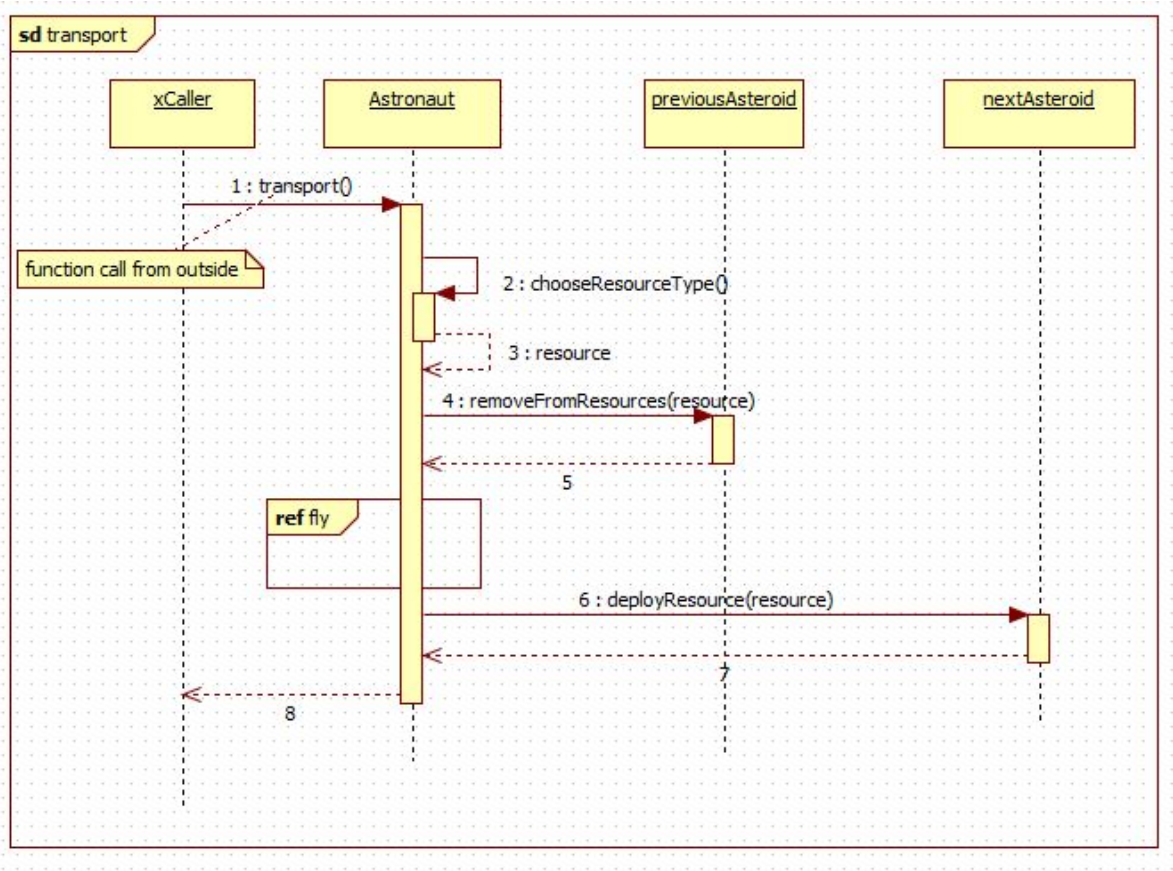
5.1.2. bore



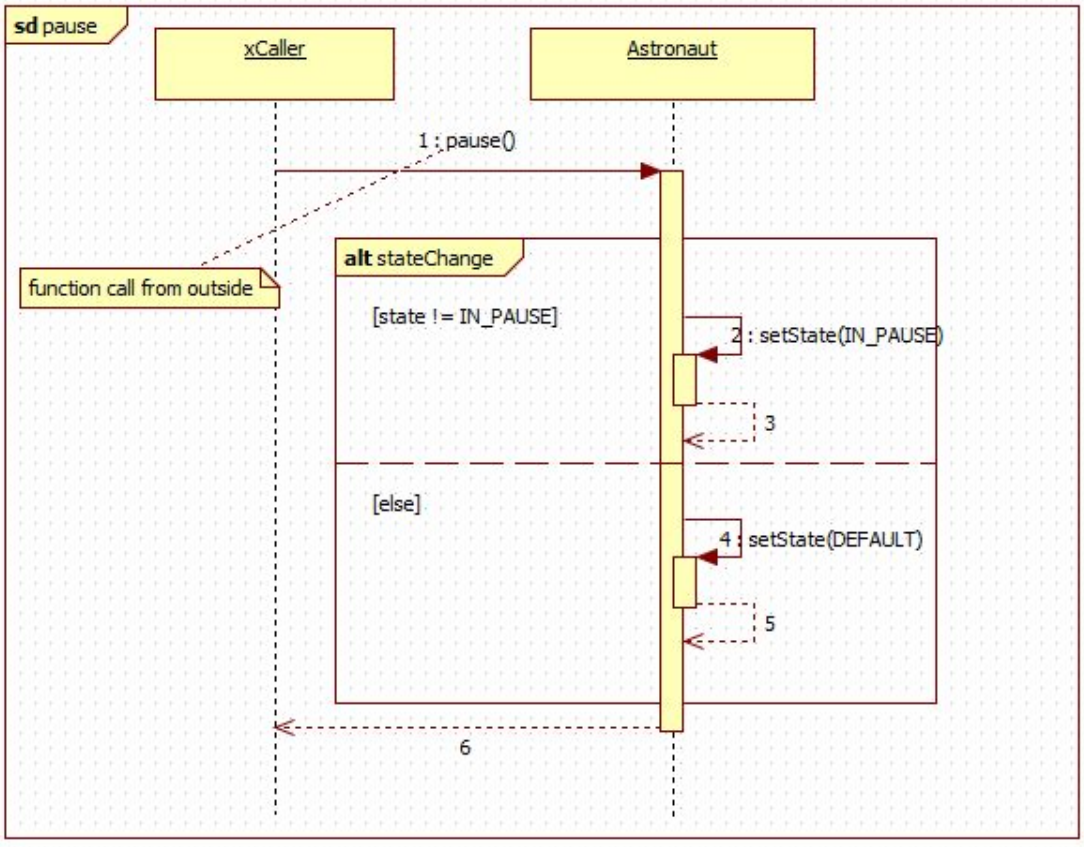
5.1.3 shield



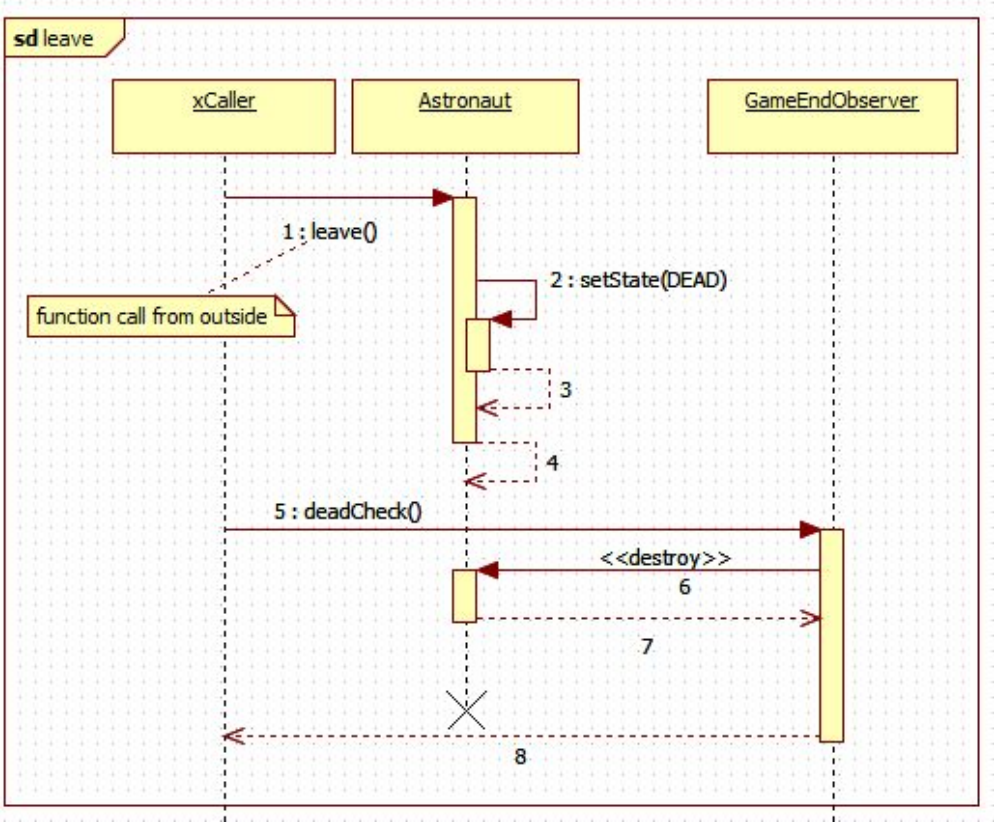
5.1.4 transport



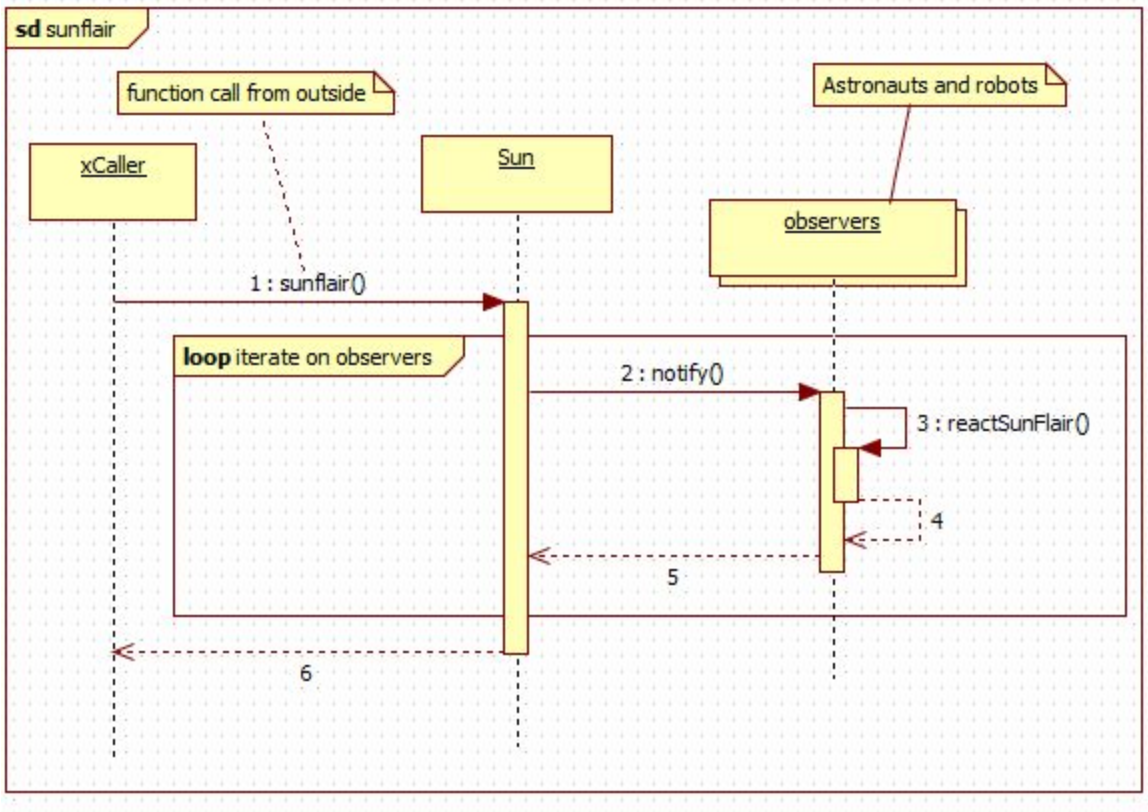
4.1.5 pause



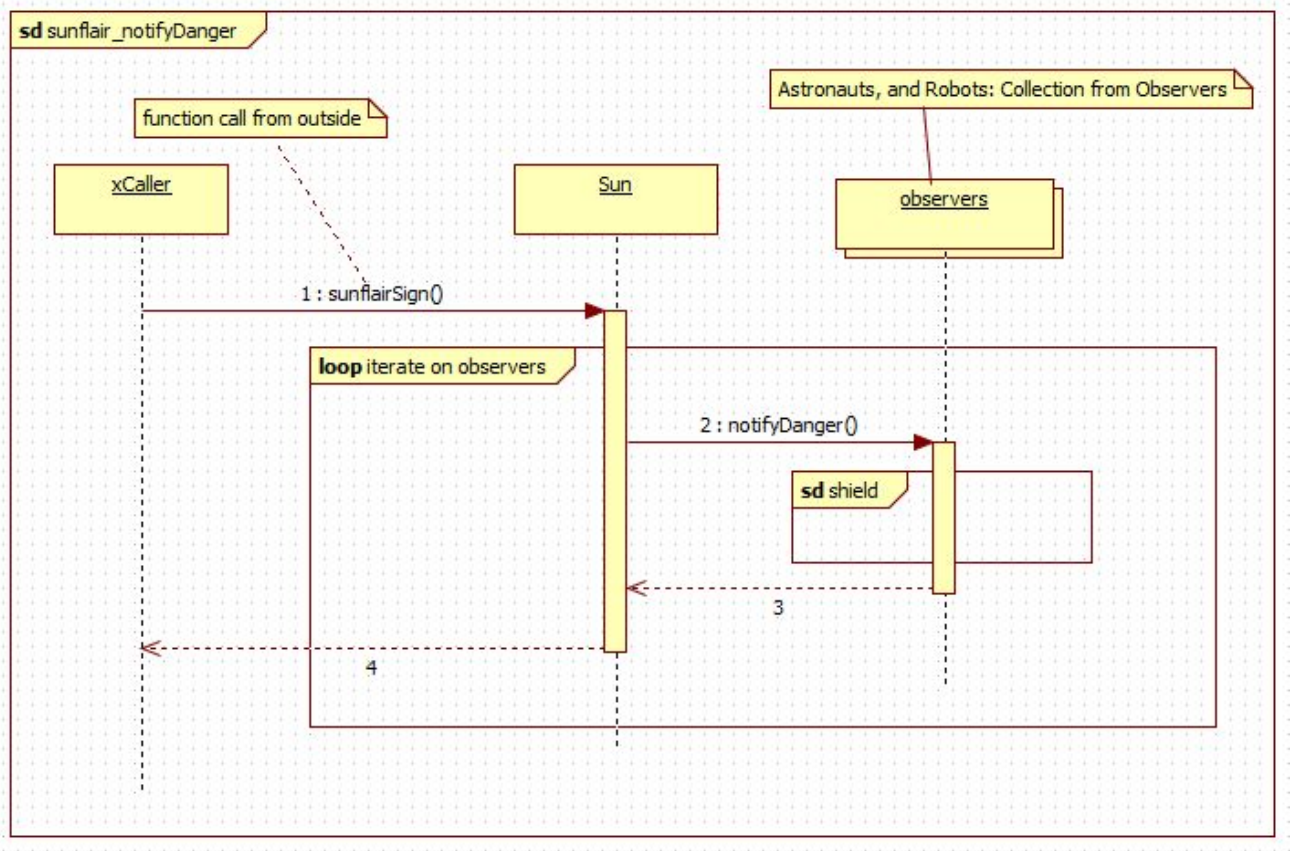
5.1.6 leave



5.1.7. sunflair_happened

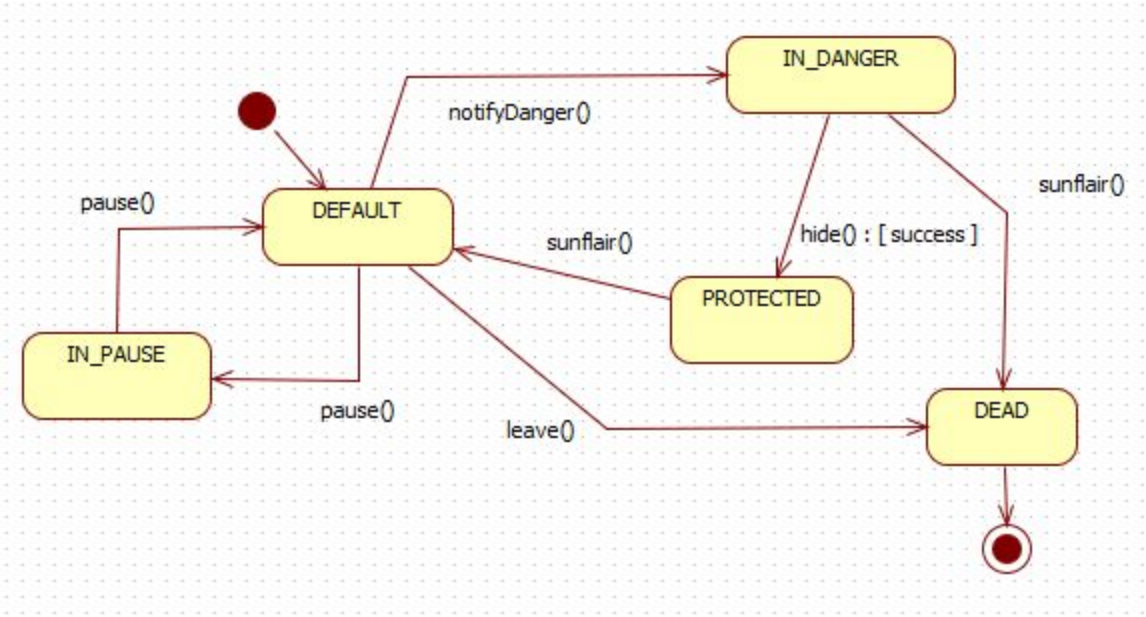


5.1.8. sunflair_notifydanger



●.2 Zustandsdiagramme

5.2.1. abstractActorState



● Aktivitäten Protokoll

Start Zeitpunkt	Aufwand in Stunden	Aktivität	Verweis
11.05 16:00	4-5 Stunden	Grobe Planung	Use case, Problemebeschreibung, und funktionale Anforderungen
11.06	ganze Tag, also 6-7 Stunden	Feinere Plaung, Klassendiagramm, Sequence Diagramm	Klassendiagramm, Sequencediagramm
11.07	4-5 Stunden	weiter das obige	Klassendiagramm, Sequencediagramm
11.09	3 Stunden	Korrektion, Ende	Klassendiagramm, Sequencediagramm, Zustandsdiagramm

Arbeitsaufwand insgesamt in Stunden:

SW Werkzeug verwendet für die Modellierung: WhiteStar UML
Andere Werkzeuge, und Ziel ihrer Anwendung: keine