Beispielszenario für die Modellierung vom Roboter

**Beschrieb der Situation Prototyp:**

Ein Roboter wird beim Spielstart im Raum C initialisiert. Der Raum C ist mit den anderen Räumen verbunden und verschlossen und nicht vom Spieler begehbar. Der Roboter ist im Moment noch ausgeschalten und somit im Power:Off Status. Im Level an der Wand ist ein Schalter X1 angebracht. Dieser schaltet zwischen den Roboter-Status On und Off hin und her.

Die Simulation beginnt damit, dass der Spieler den Schalter betätigt.

Der Roboter wechselt dann in die Aktiv – Zustand.

Er geht seinen zwei Haupttätigkeiten nach: Er schaut priorisierend darauf, das sein Akkuladestand (Powerlevel) über 30% liegt. Wenn nicht, wird der Roboter versuchen, den Akku aufzuladen.

Ist der Akku über 30%, patrouilliert der Roboter die Umgebung vom Room C.

Der Akkuzustand nimmt kontinuierlich ab, was

Es gibt also 4 Zustände:

1. Der Roboter hat gar kein Strom und ist Offline
2. Der Roboter hat Strom und läd sich auf bis..
   1. Der Roboter ist unter 30% und sucht nach einer Akku-Ladestation
   2. Der Roboter ist über 30% und patrouilliert die Umgebung

Roboter

Power:On

false

true

Active

Inactive

Powerlevel < 30%

false

true

Recharging

Patrol

Blau = Roboter-States

Rot = Hauptstates

Grün = Sub-States

**Handlungs:**

**Robot1*.State<Actor>.hasElectricity-> false***

**;** // Der Roboter ist ausgeschalten.

Wird jetzt der Strom im Raum für den Roboter verfügbar, wird dieser per Telegram benachrichtigt.

Bei der nächsten Prüfung wird festgestellt

**Robot1*.State<Actor>.hasElectricity-> true***

**;**

Und der Roboter wechselt in seinen letzten State: Er prüft also wie viel Akku er noch hat:

**Robot1. *currentState.getPowerlevel()*;**

Ist das Resultat der Prüfung über 30% wird zum Patrol State gewechselt.

**Robot1.setPreviousState *= currentState;***

**Robot1.*setCurrentState = Patrol*;**

Ist das Resultat der Prüfung unter 30% wird zum Recharching State gewechselt.

**Robot1.setPreviousState *= currentState;***

**Robot1.*setCurrentState = Recharging*;**

Jeder Spielzustandsupdate verbraucht einen gewissen Akku-Energie;

**Robot1.setPowerlevel(X);**

Zusätzlich wird immer zuerst der GlobalState geprüft, ob **hasElectricity = true** ist.

Wenn das nicht der Fall ist, wechselt der auf „Inactive“

**Robot1.setPreviousState *= currentState;***

**Robot1.*setCurrentState = Inactive*;**

Gerade der GlobalState kann via Telegram die Electricity setzen, diese wiederum beeinflusst den Roboter.