Selbstorganisierende Systeme – 2.Übung

Christian Gruber, 0625102 und Johannes Reiter, 0625101

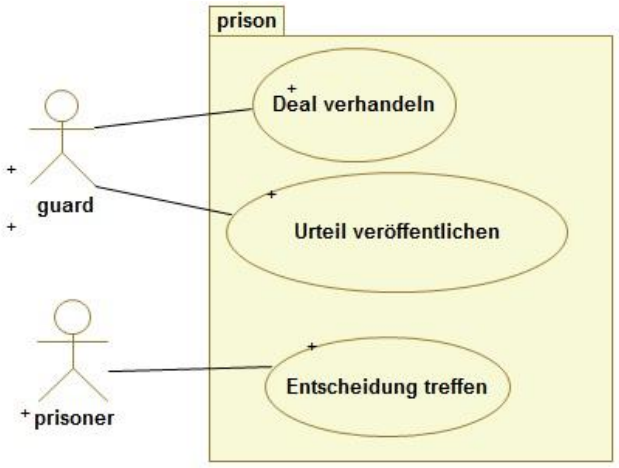
Gefangenendilemma

1. Beschreibung

Das Gefangenendilemma ist ein klassisches, symmetrisches „Zwei-Personen-Nicht-Nullsummen-Spiel“ bei dem zwei Gefangene in einem sozialen Dilemma stehen und individuell rationale Entscheidungen zu treffen haben, womit es sich perfekt für ein Multiagenten System eignet. Der Hintergrund zu diesem Spiel ist, dass es gegen zwei Gefangene nicht ausreichend Beweise gibt, um sie für ihr Verbrechen vollständig zur Verantwortung ziehen zu können. Deshalb wird ihnen getrennt von einander und ohne der Möglichkeit einer Absprache ein Deal angeboten, dass wenn sie den Partner verraten würden, frei kommen werden. Die Herausforderung bei der Findung einer guten Strategie ist, dass wenn sich beide verraten, das insgesamt schlechteste Ergebnis zustande kommt und wenn beide schweigen, das insgesamt beste Ergebnis produziert wird. Die Qualität einer Strategie kann nun über der Wiederholung des Spiels herausgefunden werden, wobei die beiden Gefangenen die Entscheidungen und Urteile der vorhergegangen Runden verwenden können.

Wir haben dieses Beispiel nun so umgesetzt, dass wir drei Agenten haben, ein Kriminalist und die beiden Gefangenen. Da das Interessante an dem Beispiel ja auch die verschiedenen Strategien sind, haben wir sieben verschiedene implementiert und zwar: Tit for Tat, Mistrust, Spite, Punisher, Pavlov, Defect, Cooperate und Random. Zur Kommunikation dieser Agenten werden nur ACL-Messages verwendet. Beim Start des Programms konfiguriert man die gewünschte Anzahl an Runden und danach wird für beide Agenten zufällige eine Strategie ausgewählt. Ist die gewünschte Anzahl an Runden erreicht, wird das Ergebnis der beiden Agenten präsentiert.

2. Anwendungsfälle



Anwendungsfallbeschreibung

Name: Deal verhandeln
Primäre Aktoren: Guard
Vorbedingung: beide Prisoner wurden vom Guard erstellt und haben eine Strategie
Szenario:

1. Guard fordert von beiden Prisoner eine Entscheidung

Name: Urteil veröffentlichen
Primäre Aktoren: Guard
Vorbedingung: beide Prisoner haben ihre Entscheidung getroffen und an den Guard geschickt
Szenario:

1. Legt das Urteil anhand der Entscheidungen der Prisoner fest
2. Guard schickt Urteil an die Prisoner

Name: Entscheidung treffen
Primäre Aktoren: Prisoner
Vorbedingung: Guard hat die Strategie übermittelt.
Szenario:

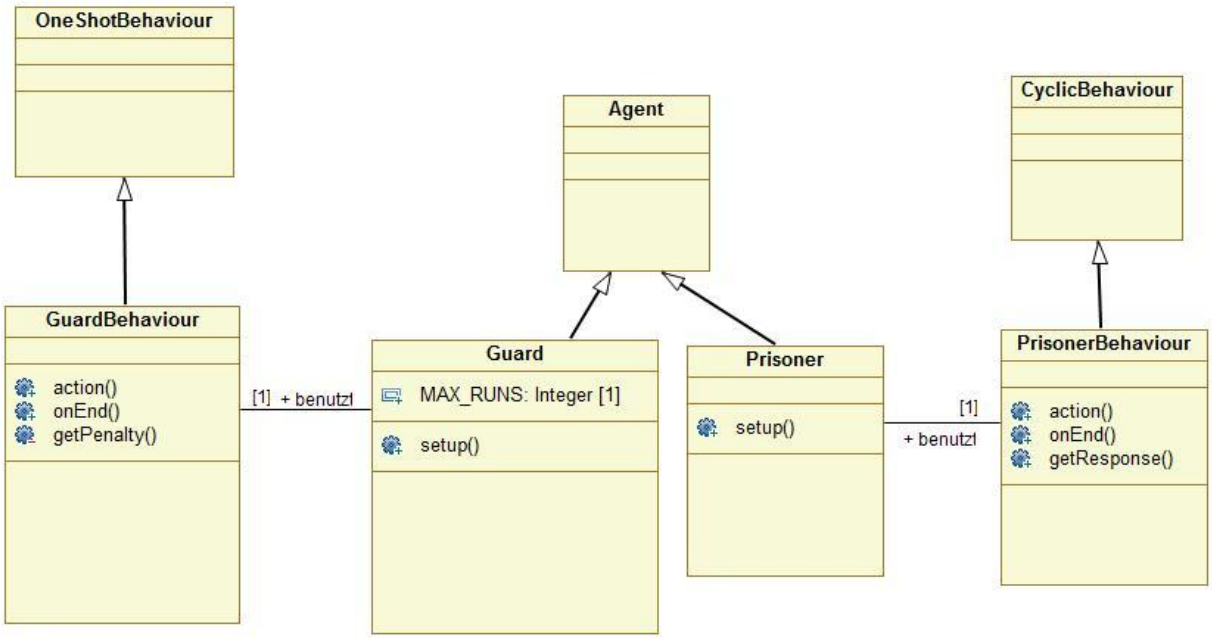
1. Trifft eigene Entscheidung
2. Schickt Entscheidung an Prisoner

3. Agentenbeschreibung

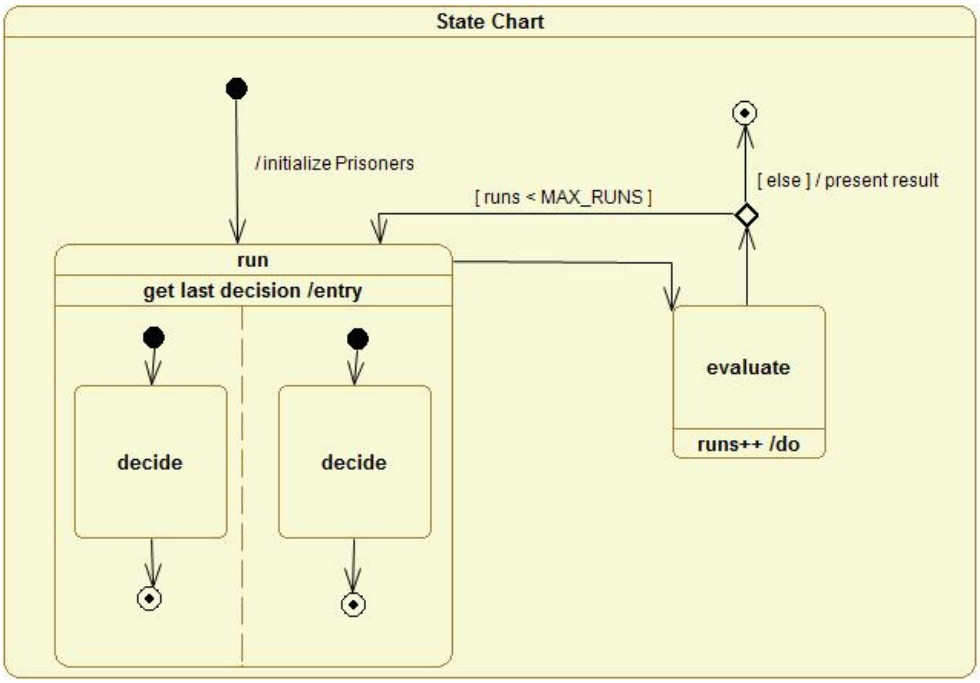
Wie zuvor kurz erwähnt gibt es in unserer Implementierung drei Agenten, einen Kriminalisten und die beiden Gefangenen.

Der Kriminalist (Guard) ist für die Verhandlungen mit den Gefangenen (Prisoner) zuständig. Ein bisschen präziser ausgedrückt implementiert er ein One-Shot-Behaviour, indem er zuerst die beiden anderen Agenten erstellt und ihnen zufällig eine Strategie zuweist. Anschließend wird noch die Konfiguration für die ACL-Messages durchgeführt und danach kann das „Spiel“ schon beginnen. In einer Schleife werden die Agenten nun nach einer Entscheidung gefragt, diese entscheiden sich je nach ihrer Strategie und geben sie dem Kriminalisten bekannt. Dafür implementieren die Gefangenen ein CyclicBehaviour, wo sie diese Aufgaben abarbeiten können. Das Urteil wird anschließend vom Kriminalisten den beiden Gefangenen übermittelt, welche dann in der nächsten Runde passend zu ihrer Strategie darauf wieder reagieren können. Wenn die gewünschte Anzahl der Runden erreicht ist, wird eine Stop-Nachricht an die beiden Agenten versendet, die sich dann beenden. Am Schluss wird noch eine Auswertung der verschiedenen Strategien der Agenten ausgegeben.

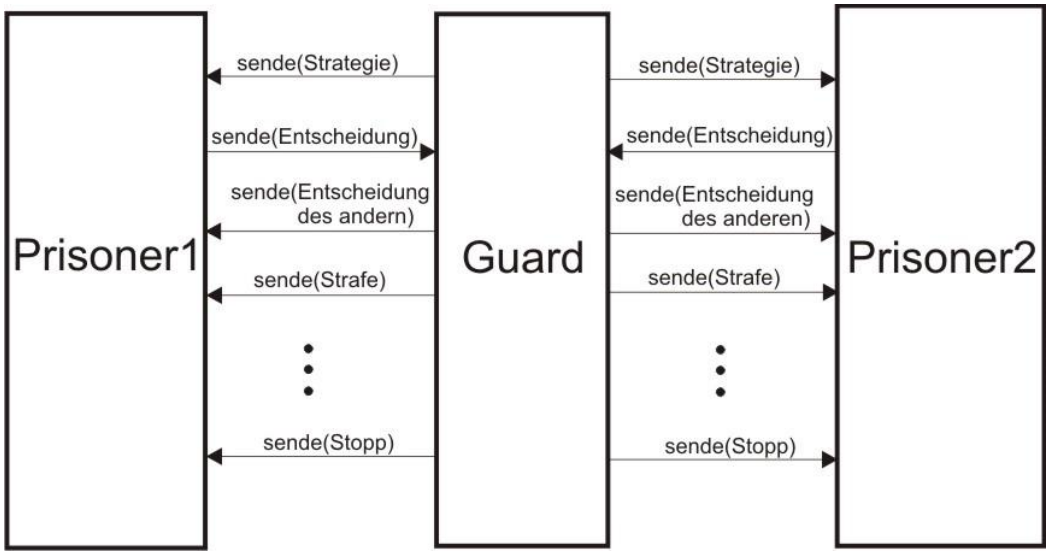
Klassendiagramm



Zustandsdiagramm



4. Nachrichten und Kommunikation



Die Kommunikation der Agenten erfolgt über ACL-Messages. Mit addReceiver wird dabei der Empfänger angegeben. Die Informationen werden dabei als Strings im Content der Message verschickt. Die beiden Prisoner kommunizieren immer nur mit dem Guard aber nie untereinander.

Innerhalb eines Runs werden mehrere Messages zwischen den Agenten verschickt.

Es werden folgenden Messages verschickt:

sende(Strategie): Zuerst sendet der Guard an die beiden Prisoner ihre Strategie (z.B. Tit for Tat, Mistrust, Spite, Punisher, Pavlov, Defect, Cooperate oder Random). Die Strategien der Prisoner können natürlich unterschiedlich sein.

sende(Entscheidung): Nachdem der Prisoner seine Entscheidung getroffen hat, schickt er diese an den Guard (z.B. HUSH - schweigen oder BETRAY - verraten).

sende(Entscheidung des Anderen): Nachdem der Guard beide Entscheidungen bekommen hat, schickt er jeweils die Entscheidung des anderen an die Prisoner (HUSH oder BETRAY).

sende(Strafe): Nachdem der Guard die Strafe der beiden festgelegt hat, schickt er die Strafen bzw. das Urteil der beiden an die Prisoner. Die Message hat die Form [eigene Strafe:Strafe des anderen]

sende(Stopp): Nachdem genügend Runs durchgeführt wurden, schickt der Guard den Stopp-Befehl an beide Prisoner damit diese terminieren. Dafür wird der String „stop“ an die Agenten geschickt.

5. Erfahrungen mit JADE

Am Anfang haben wir uns ein bisschen mit der Konfiguration des JADE-Systems herum geplagt. Bis man die richtigen Informationen und Einstellungen zusammen gesucht hat, braucht es ein wenig Zeit. Das einzig echte Problem das wir hatten war, dass unter Windows 7 JADE erst nach einem Neustart gelaufen ist. Warum und wieso – keine Ahnung. Der Rest hat eigentlich einwandfrei funktioniert, auch der Messageversand hat von Beginn an reibungslos geklappt. Vor allem mit dem JADE Graphical Tool war es recht interessant, die Abläufe beobachten zu können. Insgesamt bleibt auf jeden Fall ein positiver Eindruck von JADE hängen.