# Machine Learning

# WS2010/11

# Exercise 2

Benjamin Bachhuber (1028430)
Patrick Marschik (0625039)

January 21, 2011

## Contents

## 1 Introduction

The prisoner's dilemma is one of the fundamental problems in game theory. In the prisoners dilemma there are two prisoner's. Each of them has to decide whether to cooperate with an opponent, or defect. Both prisoners make a choice and then their decisions are revealed.

They receive a payoff according to the following table:

| Action of A / Action of B | Comply | Defect |
|---|---|---|
| Comply | 3 / 3 | 5 / 0 |
| Defect | 0 / 5 | 1 / 1 |

Since the strategies for a game with one turn or with a deterministic amount of games is fairly predictable, the so called *Iterate Prisoner's Dilemma* was used. In this special kind of Prisoner's Dilemma none of the prisoner's knows how many rounds are played.

## 2 Technical Description

One of the most important decisions is which agents are needed and what their main tasks are. The following use case diagramm helped was very useful in this process of decision making:

The use case diagram shows (see fig. 1 on page 4) that two agent classes are needed: `PrisonerAgent` (see fig. 2 on page 5) and `GamemasterAgent` (see fig. 3 on page 6).

The agents required to communicate with each other on several channels. There are different forms of communication which are required:

- The `GamemasterAgent` has to be able to ask the the PrisonerAgents if they are guilty.

- The `PrisonerAgent`s has to be able to answer if they are guilty.
- A `PrisonerAgent` has to be able to receive notifications about the game state from the `GamemasterAgent`.
- The `GamemasterAgent` has to be able to inform the `PrisonerAgent`s – that requested notifications about changes – about the current game state.

These tasks basically require two different interaction protocols (see fig. 4 on 7 on how they work together.)

**ArchieveREInitiator & ArchieveREResponder**
This protocol is used to to enable the Agents to ask and answer the question of guiltiness. Furthermore the special case of the FIPA-query protocol was used. This protocol queries an agent about what he believes about a certain state – in our case it's guiltyness.

**SubscriptionInitiator & SubscriptionResponder**
Enables the `PrisonerAgents` to register at the `GamemasterAgent` receive notification. The `GamemasterAgent` in turn sends the results of the round – i.e. what each agent voted – to all it's subscribers.

## 3 Conclusion

The prisoner's dilemma is a good example to show the strength and weaknesses of Multi Agent Systems. Using Multi Agent Systems for small projects seems like an overhead. In in large projects however Multi Agents System are a powerful architecture.

## A Diagrams

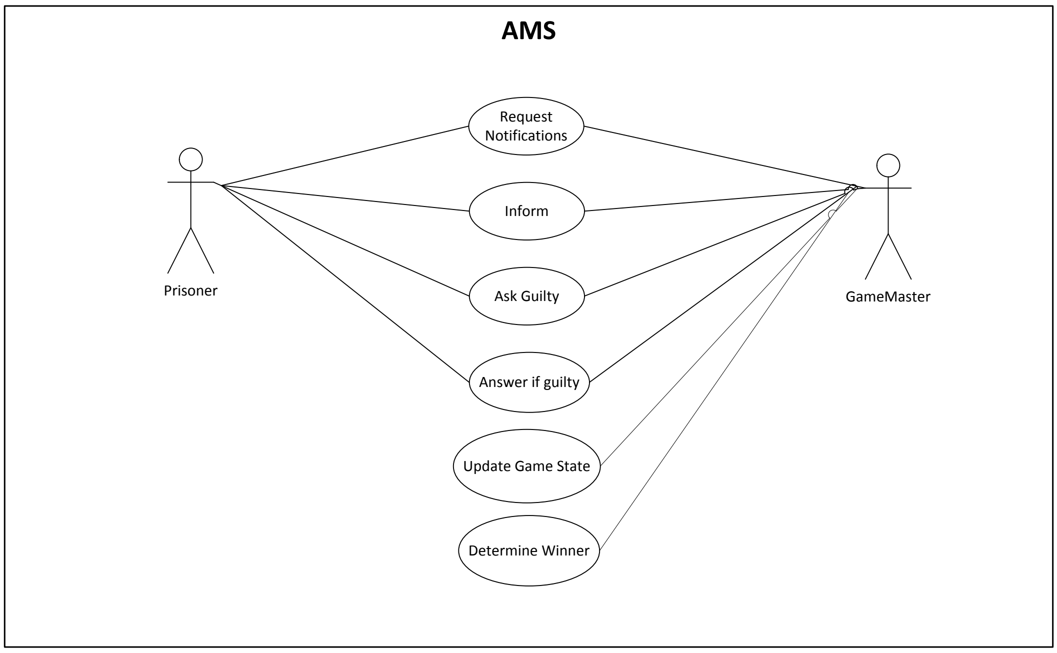

Figure 1: Use-Case diagram

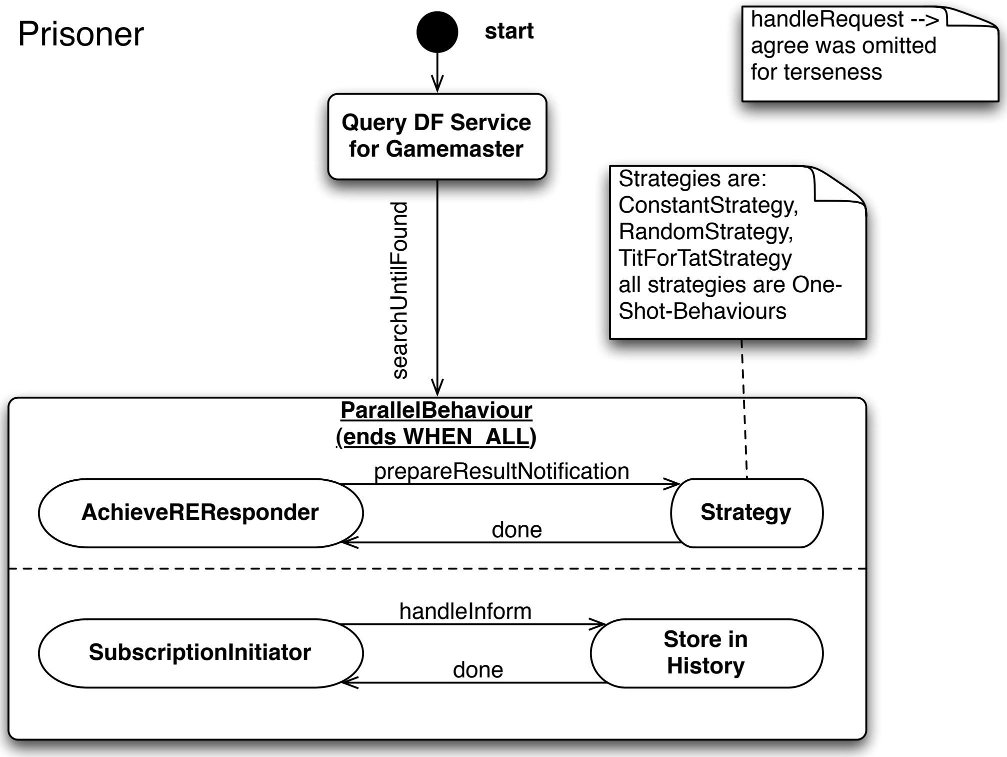
Prisoner
start
handleRequest --> agree was omitted for terseness
Query DF Service for Gamemaster
searchUntilFound
Strategies are: ConstantStrategy, RandomStrategy, TitForTatStrategy all strategies are One-Shot-Behaviours
ParallelBehaviour (ends WHEN_ALL)
prepareResultNotification
AchieveREResponder
Strategy
done
handleInform
SubscriptionInitiator
Store in History
done

Figure 2: State diagram for `PrisonerAgent`

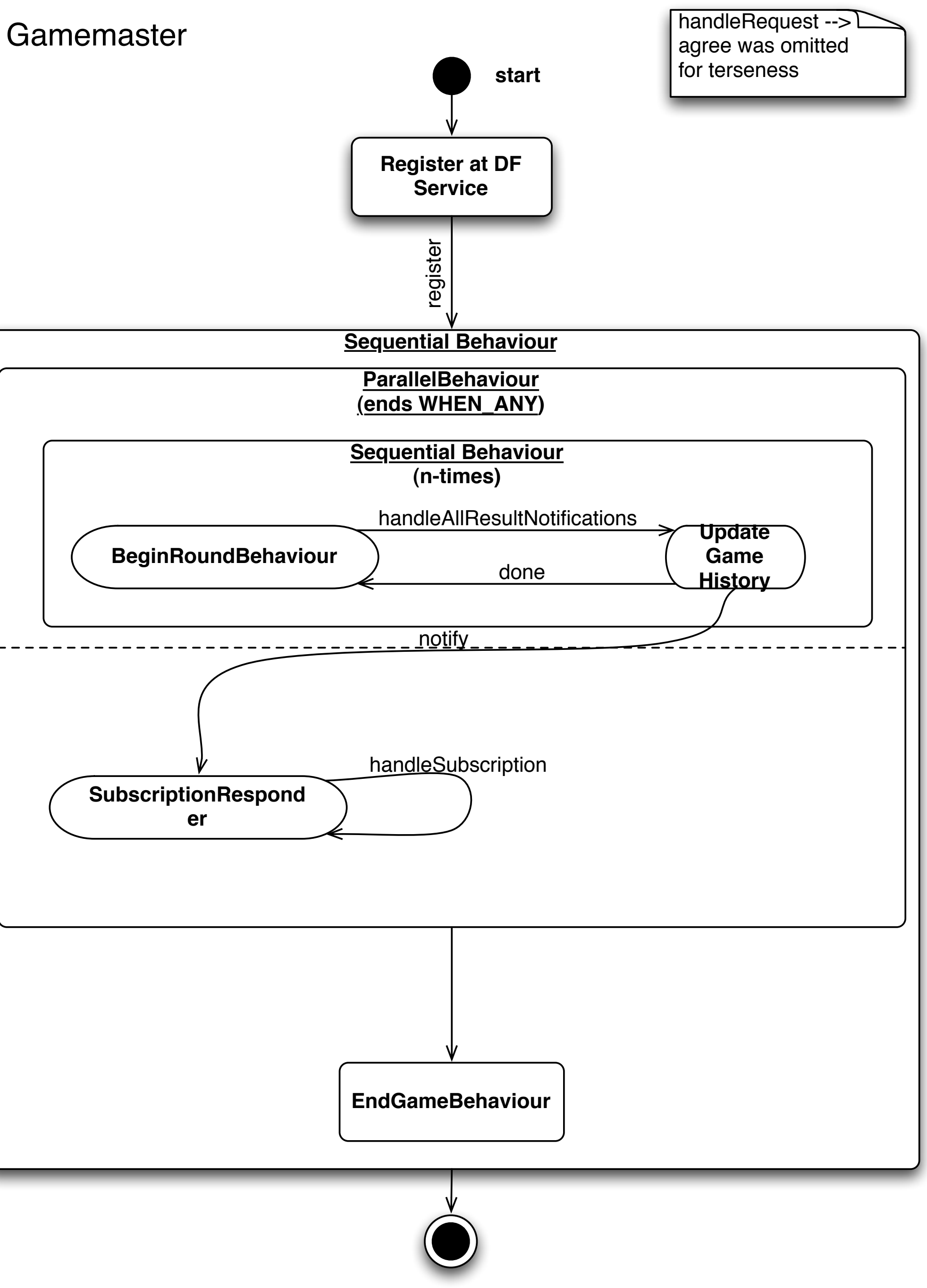

Figure 3: State diagram for `GamemasterAgent`

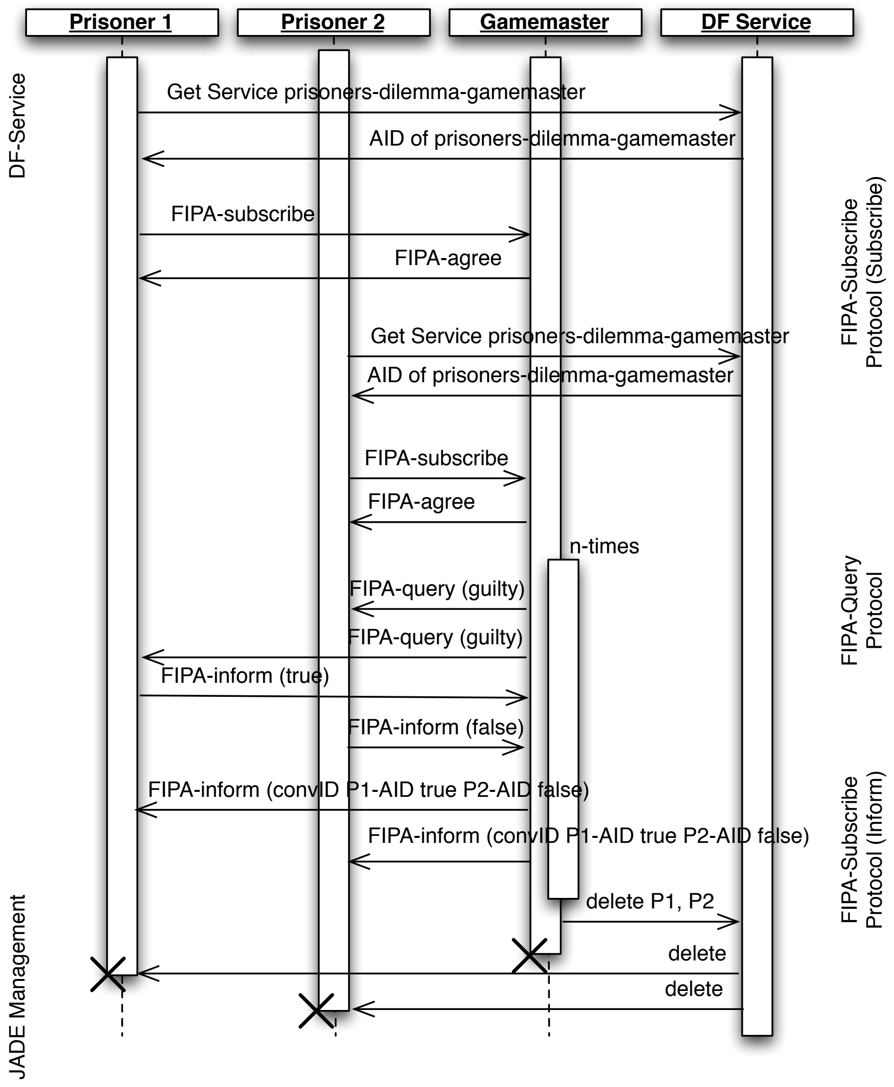
Prisoner 1
Prisoner 2
Gamemaster
DF Service
DF-Service
Get Service prisoners-dilemma-gamemaster
AID of prisoners-dilemma-gamemaster
FIPA-subscribe
FIPA-agree
Get Service prisoners-dilemma-gamemaster
AID of prisoners-dilemma-gamemaster
FIPA-subscribe
FIPA-agree
FIPA-Subscribe Protocol (Subscribe)
n-times
FIPA-query (guilty)
FIPA-query (guilty)
FIPA-inform (true)
FIPA-inform (false)
FIPA-Query Protocol
FIPA-inform (convID P1-AID true P2-AID false)
FIPA-inform (convID P1-AID true P2-AID false)
FIPA-Subscribe Protocol (Inform)
delete P1, P2
delete
delete
JADE Management

Figure 4: Sequence diagram