

# Konzeptpapier

Holger Handel, Markus Hildebrand,  
Daniel Hoffmann, Johannes Holzer, Christian Leber,  
Bastian Trauter, Andreas Zeiser

21. Juni 2002

**Inhaltsverzeichnis**

<b>1</b>	<b>Geplante Architektur</b>	<b>2</b>
1.1	Terminologie . . . . .	2
1.2	Die 11 Spieler . . . . .	2
1.3	Der einzelne Spieler . . . . .	2
<b>2</b>	<b>Verhalten/Strategie der Spieler</b>	<b>2</b>
2.1	Funkionalitäten, die als notwendig erscheinen . . . . .	2
2.2	Vermischte Notizen zum Thema Verhalten/Strategie . . . . .	3
<b>3</b>	<b>Aufteilung</b>	<b>3</b>
<b>4</b>	<b>Zeitplanung</b>	<b>3</b>

# 1 Geplante Architektur

## 1.1 Terminologie

Im folgenden wird die Java Terminologie verwendet, mit der wirklichen Implementation hat dies nicht viel zu tun, da z.B. auf Plattformen wie GNU/Linux keine native Thread Implementation verfuegbar ist. Desweiteren wird beim Design davon ausgegangen, dass keine sog. Green-Threads von der JVM verwendet werden.

## 1.2 Die 11 Spieler

Alle 11 Spieler sollen durch einen Prozess gestartet werden, dies ermöglicht auch, dass die GUI, die auch als Thread innerhalb dieses Prozesses läuft eventuell zwischen den verschiedenen Spielern umschalten kann. Die Datenstrukturen der Spieler bleiben auch bei diesem Ansatz natürlich getrennt, wie es für einen fairen Wettkampf geboten ist.

## 1.3 Der einzelne Spieler

Jeder Spieler wird aus 3 Threads bestehen (siehe auch Diagramm):

1. **Sensor:** Empfängt die Daten vom Server, parst sie und fügt diese Informationen in das Weltbild des Spielers ein.
2. **Actor:** Hauptthread: Startet die beiden anderen Threads, initialisiert das Weltbild und das Comm-Objekt und meldet den Spieler am Server an.
3. **Sender:** Sendet alle 100ms den Spielzug ab, der von der Taktik Methode des Actors zu diesem Zeitpunkt als optimal angesehen wurde.

Die Gründe für diese Architektur: Durch die Verwendung von 3 Threads pro Spieler hoffen wir einigen Problemen aus dem Weg gehen zu können, die sich mit anderen Modellen ergeben würden. Die Vorteile die wir erwarten:

- **Übersichtlichkeit:** klar abgetrennte Aufgaben für die einzelnen Threads
- **Aktualität:** alle Informationen vom Server werden zügig in das Weltbild des Spielers übernommen
- **Timing:** da die einzige Aufgabe des Sender Threads das Abschicken von Nachrichten an den Server ist, haben wir guten Grund zur Annahme, dass es mit dem Timing besser klappt, als wenn z.B. select verwendet werden würde.

# 2 Verhalten/Strategie der Spieler

Dazu kann die Gruppe zum momentanen Zeitpunkt keine genaue Aussage machen, allerdings lassen sich ein paar Dinge vorwegnehmen:

## 2.1 Funktionalitäten, die als notwendig erscheinen

Eventuell wird es hier eine Teilung in mehrere Ebenen geben, die hängt aber vom Fortschritt der Entwicklung ab.

- **LOW\_MOVE:** Mit Drehungen und Schritten direkt zu einer Stelle, wobei der Spieler anderen Objekten ausweicht.
- **LOW\_KICK:** Den Ball mit kleinen Kicks an eine bestimmte Position bringen.

- **Verteidigung:** Verteidigende Spieler stehen vor dem Tor, berechnen die Flugbahn des Balles und versuchen ihn aufzuhalten.
- **Pass:** Den Ball zu einem Mitspieler passen, wenn dies sinnvoll erscheint. (Entfernung, Gegner)

## 2.2 Vermischte Notizen zum Thema Verhalten/Strategie

- keine genetischen Algorithmen
- wohl keine selbstlernenden Algorithmen
- eventuell die Verwendung von Rufen zur Interaktion zwischen den einzelnen Spielern
- Spieler werden versuchen den Ball in das gegnerische Tor zu schießen
- Spieler werden auf ihre Stamina achten
- Spieler werden verschiedene Funktionen haben (Torwart/Verteidigung/Stürmer)
- wahrscheinlich kein Coach

## 3 Aufteilung

Da die Übergänge fließend sind, gibt es nur eine grobe Einteilung, die innerhalb der Gruppe geändert wird, falls das primäre Ziel dies erfordert. Momentan ergibt sich folgende Aufteilung, in der ungefähren Reihenfolge der Fertigstellung:

- **Pflichtenheft/Konzept:** Christian Leber
- **Weltbild:** Johannes Holzer
- **Sensor:** Markus Hildebrand
- **Sender, Hauptprogramm, Comm-Objekt:** Andreas Zeiser
- **GUI:** Daniel Hoffmann
- **Strategiekomponenten:** Bastian Trauter, Holger Handel

Aber das ändert sich natürlich entsprechend wenn einzelne Aufgaben abgeschlossen sind.

## 4 Zeitplanung

Die Zeitplanung wird sich im wesentlichen natürlich an die Vorgaben halten, wobei sich einige Gruppenmitglieder allerdings eine etwas schnellere Implementierung der low-level Funktionalitäten (Netzwerk, Threads, Datenstrukturen) wünschen um mehr Zeit für die high-level Funktionalitäten, die Feinabstimmung und natürlich das Zeichnen wunderschöner UML Diagramme zu haben.