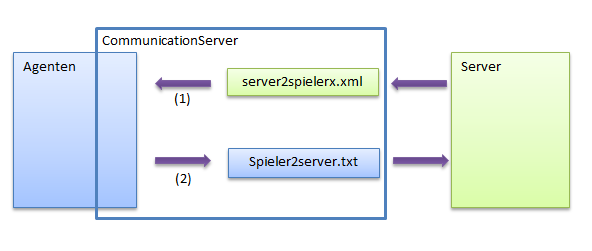
## Serverkommunikation

Dieses Paket enthält die notwendigen Bestandteile zu Kommunikation mit dem Server und somit zum Spielbetrieb über das Netzwerk.

### Aufgaben

Dieses Paket erfüllt zwei Hauptaufgaben, die in der unten stehenden Abbildung veranschaulicht werden. Links ist die Agentensoftware zu sehen und rechts der Server. In der Mitte befinden sich die Dateien zum Austausch von Informationen zwischen Agent und Server. Dabei beschreibt die Farbe der Datei, von wem diese geschrieben wird.

#### (1)Überwachung der Serverdatei für den Agenten

Der Agent überwacht in einem auf der GUI einzustellenden Frequenz das angegebene Verzeichnis, ob eine neue Datei vom Server geschrieben wurde. Dabei ist zu beachten, dass nur neue Dateien gelesen werden und keine bereits gelesenen Dateien nochmals eingelesen werden.

Wenn eine neue Datei erkannt wird, startet das Parsen dieser XML Datei. Das Ergebnis ist eine Objekt der Klasse ServerMessage, dass alle Informationen der Serverdatei enthält. Im Anschluss werden diese Informationen überprüft und entsprechende Events zur weiteren Verarbeitung werden erzeugt.

Liste der möglichen Events:

* OppMove: Zug des Gegners inklusive der jeweiligen Spalte (erster Spielzug mit Spaltenwert -1 wird ebenfalls als OppMove abgebildet)
* WinnerSet: Ein Gewinner ist vom Server gesetzt worden. Zusätzlich wird die Rolle des Gewinners mit übermittelt.
* EndSet: Der Satz wurde vom Server beendet.

Diese Events werden erzeugt und über den EventDispatcher (siehe Kapitel ??????????) weitergeleitet.

#### (2)Zurückschreiben der Züge des eigenen Agenten an den Server

Die zweite Aufgabe besteht in der Antwort des Agenten an den Server. Dazu wird eine Textdatei mit der Spaltennummern des Agentenzuges in das Serververzeichnis durch den Agenten geschrieben. Dabei wird nochmals überprüft, ob der zu schreibende Wert gültig ist.

#### Performanceoptimierung

Ein wichtiger Bestandteil der Überwachung ist, dass diese möglichst parallel abläuft und dem Programm die Möglichkeit gibt, Berechnungen und andere Funktionen gleichzeitig auszuführen. Dazu wird die erste Funktion, das Überwachen der Serverdatei, durch einen eigenen Thread realisiert. Die dazu notwendige Implementierung wurde in der Klasse ReadServerFileThread durchgeführt. Der Start ist auch im Sequenzdiagramm „Satz starten“ zu sehen.

#### Bestandteile

Für die Aufgaben der Serverkommunikation kommen folgende Klassen zum Einsatz:

1. CommunicationServer

Singleton, der den anderen Paketen als zentrale Schnittstelle dient. Über das Objekt wird die Serverdatei Überwachung gestartet und kann jederzeit wieder beendet werden. Die Klasse leitet die Informationen der Serverdatei über Events an das Hauptprogramm weiter. Des Weiteren verwaltet das Objekt die Pfade zu den Dateien und die Dateinamen, die entsprechend der eigenen Rolle definiert werden. Außerdem wird die Agenten Datei an den Server durch diese Klasse geschrieben,

Eine weitere Funktion ist das Testen der Schreibgeschwindigkeit. Damit soll sichergestellt werden, dass die Timeoutzeiten nicht durch einen schlechten Netzwerkzugriff überschritten werden.

1. ReadServerFileThread

Diese Klasse dient ausschließlich dazu, die Überwachung der Serverdatei als eigenen Thread zu erzeugen um eine parallele Abarbeitung zu ermöglichen.

1. XMLParser

Das Parsen und auswerten des Serverfiles für die Bereitstellung der Informationen erfolgt in dieser Klasse. Sie ist ebenfalls als Singleton realisiert. Sie liest das Serverfile aus, parst das XML und erstellt, soweit die XML wohl geformt ist, eine ServerMessage Instanz.

1. ServerMessage

Dieses Objekt bietet die Datenpersistenz für die Nachrichten vom Server an den Agenten. Für jede Servernachricht wird eine Instanz erzeugt.