

Einführungsvortrag zur Diplomarbeit
„Repräsentation und Simulation von Umgebungen
für Multiagentensysteme“

Stefan Tittel

Technische Universität Dortmund

6. Mai 2010

- 1 Einführung
 - Definitionen
 - Beispiel
- 2 Problemstellung
 - Überblick
 - Festlegung der Eigenschaften und Funktionen
 - Zustandsbeschreibung und -überführung
 - Spezifikationsformat
 - Kommunikationsprotokolle
 - Implementierung
- 3 Zusammenfassung

- 1 Einführung
 - Definitionen
 - Beispiel
- 2 Problemstellung
 - Überblick
 - Festlegung der Eigenschaften und Funktionen
 - Zustandsbeschreibung und -überführung
 - Spezifikationsformat
 - Kommunikationsprotokolle
 - Implementierung
- 3 Zusammenfassung

Definition: Agent und Multiagentensystem

Was ist ein Agent?

- keine allgemeinverbindliche Definition
- zentrale Eigenschaft: Autonomie

Definition nach Wooldridge

An **agent** is a computer system that is **situated** in some **environment**, and that is capable of **autonomous action** in this environment in order to meet its design objectives.

Was ist ein Multiagentensystem?

Definition nach Krupansky

A **multi-agent system (MAS)** is a computational system where **agents** cooperate or compete with others to achieve some individual or collective task.

Definition: Umgebung

unterschiedliches Verständnis des Begriffs „Umgebung“, z. B.:

- Kommunikationsinfrastruktur für Agenten
- Software-Infrastruktur, die Agenten ausführt
- Hardware-Infrastruktur für das Multiagentensystem
- reale Welt

hier:

Definition

Eine **Umgebung** stellt Agenten eine **virtuelle Welt** bereit. Der Zustand dieser Welt wird Agenten (ggf. nur ausschnittsweise) durch Nachrichten übermittelt. Agenten führen Aktionen in der Welt durch den Versand von Nachrichten an die Umgebung durch.

Staubsaugerwelt

- Welt besteht aus Grid der Größe 6×3
- Objekttypen: Agenten, Staubknäuel
- jedem Objekt ist genau eine Grid-Zelle zugeordnet
- Kapazität der Grid-Zellen unbeschränkt
- mögliche Aktionen:
 - Staubknäuel aufsaugen (erfolgreich, wenn Agent in gleicher Grid-Zelle wie Staubknäuel)
 - Bewegung um eine Grid-Zelle nach Norden, Osten, Süden oder Westen (erfolgreich, wenn Ziel nicht außerhalb des Grids)

Beispiel für eine Umgebung 2/2

2		Agent	Staub			Staub
1						
0					Staub	
	0	1	2	3	4	5

Nachricht: Umgebung \rightarrow Agent

Grid-Dimension: 6×3 , Staubknäuel: $(2, 2)$, $(4, 0)$, $(5, 2)$, Agent: $(1, 2)$

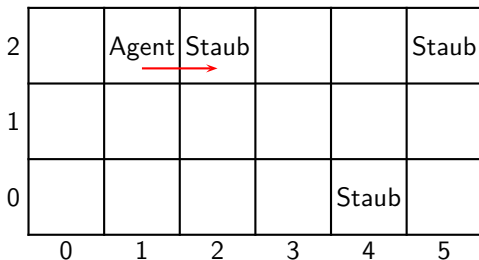
Nachricht: Agent \rightarrow Umgebung

Ich gehe nach Osten.

Nachricht: Umgebung \rightarrow Agent

Grid-Dimension: 6×3 , Staubknäuel: $(2, 2)$, $(4, 0)$, $(5, 2)$, Agent: $(2, 2)$

Beispiel für eine Umgebung 2/2



Nachricht: Umgebung \rightarrow Agent

Grid-Dimension: 6×3 , Staubknäuel: $(2, 2)$, $(4, 0)$, $(5, 2)$, Agent: $(1, 2)$

Nachricht: Agent \rightarrow Umgebung

Ich gehe nach Osten.

Nachricht: Umgebung \rightarrow Agent

Grid-Dimension: 6×3 , Staubknäuel: $(2, 2)$, $(4, 0)$, $(5, 2)$, Agent: $(2, 2)$

Beispiel für eine Umgebung 2/2

2		Agent Staub			Staub	
1						
0				Staub		
	0	1	2	3	4	5

Nachricht: Umgebung \rightarrow Agent

Grid-Dimension: 6×3 , Staubknäuel: $(2, 2)$, $(4, 0)$, $(5, 2)$, Agent: $(1, 2)$

Nachricht: Agent \rightarrow Umgebung

Ich gehe nach Osten.

Nachricht: Umgebung \rightarrow Agent

Grid-Dimension: 6×3 , Staubknäuel: $(2, 2)$, $(4, 0)$, $(5, 2)$, Agent: $(2, 2)$

- 1 Einführung
 - Definitionen
 - Beispiel
- 2 Problemstellung
 - Überblick
 - Festlegung der Eigenschaften und Funktionen
 - Zustandsbeschreibung und -überführung
 - Spezifikationsformat
 - Kommunikationsprotokolle
 - Implementierung
- 3 Zusammenfassung

Ziel der Diplomarbeit: Konzeption und Implementierung eines flexiblen Frameworks zur Realisierung von Umgebungen für Multiagentensysteme

Aufteilung in Unterziele:

- 1 Eigenschaften und Funktionen zu unterstützender Umgebungen festlegen
- 2 Zustandsbeschreibungen und -überführungen von Umgebungszuständen erarbeiten
- 3 Spezifikationsformat für Umgebungen entwickeln
- 4 Entwicklung von Protokollen zur Kommunikation zwischen Agenten und Umgebung
- 5 Implementierung durchführen

- vollständig wahrnehmbar oder partiell wahrnehmbar
- deterministisch oder stochastisch
- episodisch oder sequentiell
- statisch oder dynamisch
- diskret oder kontinuierlich
- reaktiv oder proaktiv
- ortsbezogen oder nicht ortsbezogen

- vollständig wahrnehmbar oder **partiell wahrnehmbar**
- deterministisch oder stochastisch
- episodisch oder sequentiell
- statisch oder dynamisch
- diskret oder kontinuierlich
- reaktiv oder proaktiv
- ortsbezogen oder nicht ortsbezogen

- vollständig wahrnehmbar oder **partiell wahrnehmbar**
- deterministisch oder stochastisch
- episodisch oder **sequentiell**
- statisch oder dynamisch
- diskret oder kontinuierlich
- reaktiv oder proaktiv
- ortsbezogen oder nicht ortsbezogen

- vollständig wahrnehmbar oder **partiell wahrnehmbar**
- deterministisch oder stochastisch
- episodisch oder **sequentiell**
- statisch oder **dynamisch**
- diskret oder kontinuierlich
- reaktiv oder proaktiv
- ortsbezogen oder nicht ortsbezogen

- vollständig wahrnehmbar oder **partiell wahrnehmbar**
- deterministisch oder stochastisch
- episodisch oder **sequentiell**
- statisch oder **dynamisch**
- **diskret** oder kontinuierlich
- reaktiv oder proaktiv
- ortsbezogen oder nicht ortsbezogen

- vollständig wahrnehmbar oder **partiell wahrnehmbar**
- deterministisch oder stochastisch
- episodisch oder **sequentiell**
- statisch oder **dynamisch**
- **diskret** oder kontinuierlich
- reaktiv oder proaktiv
- **ortsbezogen** oder nicht ortsbezogen

- zweidimensionales Grid als zentrale räumliche Komponente
- Nachrichtenvermittlung zwischen Agenten
 - jede Nachricht zwischen Agenten über die Umgebung
 - Kapselung beliebiger Nachrichteninhalte
 - Einschränkung der Empfangsreichweite
- Interaktion mit Objekten
 - Aufnahme, Verschiebung, Ablage, Weitergabe
 - zu diskutieren: Interaktion mit komplexen Objekten
- Einschränkung der Bewegungs- und Wahrnehmungsmöglichkeiten
 - Hindernisse
 - maximale Sichtreichweite
- Synchronisation und Ausführungsreihenfolge
- Konfigurierbarkeit
 - Grid: z. B. Größe des Grids, Größe der Zellen
 - Objekte: z. B. Objekttypen, Objektgewichte
 - Agenten: z. B. erlaubte Aktionen, maximale Traglast

- zweidimensionales Grid als zentrale räumliche Komponente
- Nachrichtenvermittlung zwischen Agenten
 - jede Nachricht zwischen Agenten über die Umgebung
 - Kapselung beliebiger Nachrichteninhalte
 - Einschränkung der Empfangsreichweite
- Interaktion mit Objekten
 - Aufnahme, Verschiebung, Ablage, Weitergabe
 - zu diskutieren: Interaktion mit komplexen Objekten
- Einschränkung der Bewegungs- und Wahrnehmungsmöglichkeiten
 - Hindernisse
 - maximale Sichtreichweite
- Synchronisation und Ausführungsreihenfolge
- Konfigurierbarkeit
 - Grid: z. B. Größe des Grids, Größe der Zellen
 - Objekte: z. B. Objekttypen, Objektgewichte
 - Agenten: z. B. erlaubte Aktionen, maximale Traglast

- zweidimensionales Grid als zentrale räumliche Komponente
- Nachrichtenvermittlung zwischen Agenten
 - jede Nachricht zwischen Agenten über die Umgebung
 - Kapselung beliebiger Nachrichteninhalte
 - Einschränkung der Empfangsreichweite
- Interaktion mit Objekten
 - Aufnahme, Verschiebung, Ablage, Weitergabe
 - zu diskutieren: Interaktion mit komplexen Objekten
- Einschränkung der Bewegungs- und Wahrnehmungsmöglichkeiten
 - Hindernisse
 - maximale Sichtreichweite
- Synchronisation und Ausführungsreihenfolge
- Konfigurierbarkeit
 - Grid: z. B. Größe des Grids, Größe der Zellen
 - Objekte: z. B. Objekttypen, Objektgewichte
 - Agenten: z. B. erlaubte Aktionen, maximale Traglast

- zweidimensionales Grid als zentrale räumliche Komponente
- Nachrichtenvermittlung zwischen Agenten
 - jede Nachricht zwischen Agenten über die Umgebung
 - Kapselung beliebiger Nachrichteninhalte
 - Einschränkung der Empfangsreichweite
- Interaktion mit Objekten
 - Aufnahme, Verschiebung, Ablage, Weitergabe
 - zu diskutieren: Interaktion mit komplexen Objekten
- Einschränkung der Bewegungs- und Wahrnehmungsmöglichkeiten
 - Hindernisse
 - maximale Sichtreichweite
- Synchronisation und Ausführungsreihenfolge
- Konfigurierbarkeit
 - Grid: z. B. Größe des Grids, Größe der Zellen
 - Objekte: z. B. Objekttypen, Objektgewichte
 - Agenten: z. B. erlaubte Aktionen, maximale Traglast

- zweidimensionales Grid als zentrale räumliche Komponente
- Nachrichtenvermittlung zwischen Agenten
 - jede Nachricht zwischen Agenten über die Umgebung
 - Kapselung beliebiger Nachrichteninhalte
 - Einschränkung der Empfangsreichweite
- Interaktion mit Objekten
 - Aufnahme, Verschiebung, Ablage, Weitergabe
 - zu diskutieren: Interaktion mit komplexen Objekten
- Einschränkung der Bewegungs- und Wahrnehmungsmöglichkeiten
 - Hindernisse
 - maximale Sichtreichweite
- Synchronisation und Ausführungsreihenfolge
- Konfigurierbarkeit
 - Grid: z. B. Größe des Grids, Größe der Zellen
 - Objekte: z. B. Objekttypen, Objektgewichte
 - Agenten: z. B. erlaubte Aktionen, maximale Traglast

- zweidimensionales Grid als zentrale räumliche Komponente
- Nachrichtenvermittlung zwischen Agenten
 - jede Nachricht zwischen Agenten über die Umgebung
 - Kapselung beliebiger Nachrichteninhalte
 - Einschränkung der Empfangsreichweite
- Interaktion mit Objekten
 - Aufnahme, Verschiebung, Ablage, Weitergabe
 - zu diskutieren: Interaktion mit komplexen Objekten
- Einschränkung der Bewegungs- und Wahrnehmungsmöglichkeiten
 - Hindernisse
 - maximale Sichtreichweite
- Synchronisation und Ausführungsreihenfolge
- Konfigurierbarkeit
 - Grid: z. B. Größe des Grids, Größe der Zellen
 - Objekte: z. B. Objekttypen, Objektgewichte
 - Agenten: z. B. erlaubte Aktionen, maximale Traglast

Idee: dynamische Zustände

- dynamischer Zustand $\delta \in \Delta$ ist ein Paar $\langle \sigma, \gamma \rangle$ mit Umgebungszustand $\sigma \in \Sigma$ und Einflüssen $\gamma \in \Gamma$
- Agenten und die Umgebung selbst können Einflüsse erzeugen
 - Einflüsse von Agenten erhält die Umgebung durch Nachrichten
 - Funktion **Exec**: $\Sigma \times \Gamma \rightarrow \Gamma$ für von der Umgebung erzeugte Einflüsse, z. B. Abbremsen eines sich bewegenden Objekts durch Reibung
- Funktion **React**: $\Sigma \times \Gamma \rightarrow \Sigma$ erzeugt neuen Umgebungszustand
- Zustandsüberführung von $\delta = \langle \sigma, \gamma \rangle$ nach $\delta' = \langle \sigma', \gamma' \rangle$
 - $\sigma' = \text{React}(\sigma, \gamma)$
 - $\gamma' = \text{Exec}(\sigma', \gamma)$
- Erweiterung der Funktionen:
 - **Exec**: $Op \times \Sigma \times \Gamma \rightarrow \Gamma$ mit $Op = \Sigma \rightarrow \Gamma$
 - **React**: $Laws \times \Sigma \times \Gamma \rightarrow \Sigma$ mit $\lambda \in Laws$ und λ z. B. ein 4-Tupel der Form $\langle \text{name}, \text{prestate}, \text{preinfluence}, \text{post} \rangle$

Idee: dynamische Zustände

- dynamischer Zustand $\delta \in \Delta$ ist ein Paar $\langle \sigma, \gamma \rangle$ mit Umgebungszustand $\sigma \in \Sigma$ und Einflüssen $\gamma \in \Gamma$
- Agenten und die Umgebung selbst können Einflüsse erzeugen
 - Einflüsse von Agenten erhält die Umgebung durch Nachrichten
 - Funktion **Exec**: $\Sigma \times \Gamma \rightarrow \Gamma$ für von der Umgebung erzeugte Einflüsse, z. B. Abbremsen eines sich bewegenden Objekts durch Reibung
- Funktion **React**: $\Sigma \times \Gamma \rightarrow \Sigma$ erzeugt neuen Umgebungszustand
- Zustandsüberführung von $\delta = \langle \sigma, \gamma \rangle$ nach $\delta' = \langle \sigma', \gamma' \rangle$
 - $\sigma' = \text{React}(\sigma, \gamma)$
 - $\gamma' = \text{Exec}(\sigma', \gamma)$
- Erweiterung der Funktionen:
 - **Exec**: $Op \times \Sigma \times \Gamma \rightarrow \Gamma$ mit $Op = \Sigma \rightarrow \Gamma$
 - **React**: $Laws \times \Sigma \times \Gamma \rightarrow \Sigma$ mit $\lambda \in Laws$ und λ z. B. ein 4-Tupel der Form $\langle \text{name}, \text{prestate}, \text{preinfluence}, \text{post} \rangle$

Idee: dynamische Zustände

- dynamischer Zustand $\delta \in \Delta$ ist ein Paar $\langle \sigma, \gamma \rangle$ mit Umgebungszustand $\sigma \in \Sigma$ und Einflüssen $\gamma \in \Gamma$
- Agenten und die Umgebung selbst können Einflüsse erzeugen
 - Einflüsse von Agenten erhält die Umgebung durch Nachrichten
 - Funktion **Exec**: $\Sigma \times \Gamma \rightarrow \Gamma$ für von der Umgebung erzeugte Einflüsse, z. B. Abbremsen eines sich bewegenden Objekts durch Reibung
- Funktion **React**: $\Sigma \times \Gamma \rightarrow \Sigma$ erzeugt neuen Umgebungszustand
- Zustandsüberführung von $\delta = \langle \sigma, \gamma \rangle$ nach $\delta' = \langle \sigma', \gamma' \rangle$
 - $\sigma' = \text{React}(\sigma, \gamma)$
 - $\gamma' = \text{Exec}(\sigma', \gamma)$
- Erweiterung der Funktionen:
 - **Exec**: $Op \times \Sigma \times \Gamma \rightarrow \Gamma$ mit $Op = \Sigma \rightarrow \Gamma$
 - **React**: $Laws \times \Sigma \times \Gamma \rightarrow \Sigma$ mit $\lambda \in Laws$ und λ z. B. ein 4-Tupel der Form $\langle \text{name}, \text{prestate}, \text{preinfluence}, \text{post} \rangle$

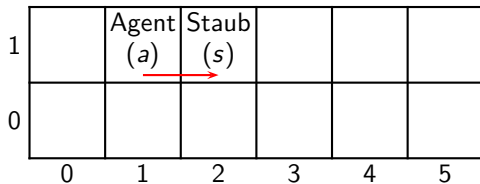
Idee: dynamische Zustände

- dynamischer Zustand $\delta \in \Delta$ ist ein Paar $\langle \sigma, \gamma \rangle$ mit Umgebungszustand $\sigma \in \Sigma$ und Einflüssen $\gamma \in \Gamma$
- Agenten und die Umgebung selbst können Einflüsse erzeugen
 - Einflüsse von Agenten erhält die Umgebung durch Nachrichten
 - Funktion **Exec**: $\Sigma \times \Gamma \rightarrow \Gamma$ für von der Umgebung erzeugte Einflüsse, z. B. Abbremsen eines sich bewegenden Objekts durch Reibung
- Funktion **React**: $\Sigma \times \Gamma \rightarrow \Sigma$ erzeugt neuen Umgebungszustand
- Zustandsüberführung von $\delta = \langle \sigma, \gamma \rangle$ nach $\delta' = \langle \sigma', \gamma' \rangle$
 - $\sigma' = \text{React}(\sigma, \gamma)$
 - $\gamma' = \text{Exec}(\sigma', \gamma)$
- Erweiterung der Funktionen:
 - **Exec**: $Op \times \Sigma \times \Gamma \rightarrow \Gamma$ mit $Op = \Sigma \rightarrow \Gamma$
 - **React**: $Laws \times \Sigma \times \Gamma \rightarrow \Sigma$ mit $\lambda \in Laws$ und λ z. B. ein 4-Tupel der Form $\langle \text{name}, \text{prestate}, \text{preinfluence}, \text{post} \rangle$

Idee: dynamische Zustände

- dynamischer Zustand $\delta \in \Delta$ ist ein Paar $\langle \sigma, \gamma \rangle$ mit Umgebungszustand $\sigma \in \Sigma$ und Einflüssen $\gamma \in \Gamma$
- Agenten und die Umgebung selbst können Einflüsse erzeugen
 - Einflüsse von Agenten erhält die Umgebung durch Nachrichten
 - Funktion **Exec**: $\Sigma \times \Gamma \rightarrow \Gamma$ für von der Umgebung erzeugte Einflüsse, z. B. Abbremsen eines sich bewegenden Objekts durch Reibung
- Funktion **React**: $\Sigma \times \Gamma \rightarrow \Sigma$ erzeugt neuen Umgebungszustand
- Zustandsüberführung von $\delta = \langle \sigma, \gamma \rangle$ nach $\delta' = \langle \sigma', \gamma' \rangle$
 - $\sigma' = \text{React}(\sigma, \gamma)$
 - $\gamma' = \text{Exec}(\sigma', \gamma)$
- Erweiterung der Funktionen:
 - **Exec**: $Op \times \Sigma \times \Gamma \rightarrow \Gamma$ mit $Op = \Sigma \rightarrow \Gamma$
 - **React**: $Laws \times \Sigma \times \Gamma \rightarrow \Sigma$ mit $\lambda \in Laws$ und λ z. B. ein 4-Tupel der Form $\langle \text{name}, \text{prestate}, \text{preinfluence}, \text{post} \rangle$

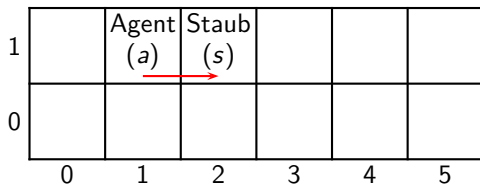
Zustandsbeschreibung und -überführung 2/2



Beispiel

$$\sigma = \{\text{Agent}(a), \text{Staub}(s), \text{Loc}(1, 1, a), \text{Loc}(2, 1, s)\} \cup \{\text{Grid}(x, y) \mid x \in \mathbb{N}^{\leq 5}, y \in \mathbb{N}^{\leq 1}\}$$

$$\lambda = \langle \text{MoveEast}(z), \{\text{Agent}(z), \text{Loc}(x, y, z), \text{Grid}(x + 1, y)\}, \{\text{StepEast}(z)\}, \{\neg \text{Loc}(x, y, z), \text{Loc}(x + 1, y, z)\} \rangle$$



Beispiel

$$\sigma = \{\text{Agent}(a), \text{Staub}(s), \text{Loc}(1, 1, a), \text{Loc}(2, 1, s)\} \cup \\ \{\text{Grid}(x, y) \mid x \in \mathbb{N}^{\leq 5}, y \in \mathbb{N}^{\leq 1}\}$$

$$\lambda = \langle \text{MoveEast}(z), \\ \{\text{Agent}(z), \text{Loc}(x, y, z), \text{Grid}(x + 1, y)\}, \\ \{\text{StepEast}(z)\}, \\ \{\neg \text{Loc}(x, y, z), \text{Loc}(x + 1, y, z)\} \rangle$$

- soll folgende Funktionen erfüllen:
 - Spezifikation des initialen Umgebungszustandes
 - Spezifikation technischer Aspekte (z. B. Serverkonfiguration, Authentifikation)
 - ggf. Spezifikation proaktiven/stochastischen Verhaltens
 - ggf. Spezifikation benutzerdefinierter Zustandsüberführungen
- Ziel: einfach, flexibel, effizient, funktionsvollständig
- Wahl der Syntax/Metasprache
- Evaluation der *Environment Description Language for Multi-Agent Simulation (ELMS)*

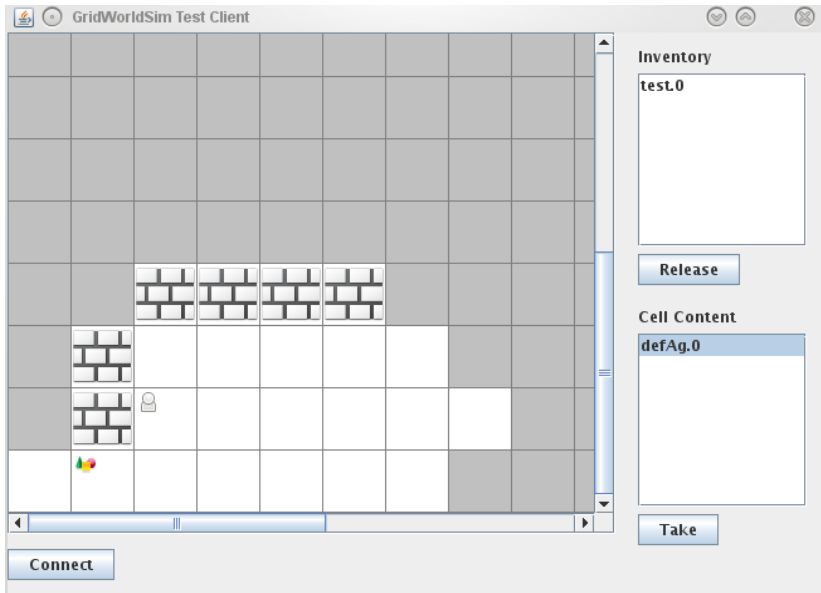
- soll folgende Funktionen erfüllen:
 - Spezifikation des initialen Umgebungszustandes
 - Spezifikation technischer Aspekte (z. B. Serverkonfiguration, Authentifikation)
 - ggf. Spezifikation proaktiven/stochastischen Verhaltens
 - ggf. Spezifikation benutzerdefinierter Zustandsüberführungen
- **Ziel:** einfach, flexibel, effizient, funktionsvollständig
- Wahl der Syntax/Metasprache
- Evaluation der *Environment Description Language for Multi-Agent Simulation (ELMS)*

- sollen folgende Funktionen erfüllen:
 - Versand von Nachrichten durch Agenten über die Umgebung
 - zu spezifischem Agenten in „Flüster-Reichweite“
 - Multicast zu allen Agenten in Reichweite
 - Information der Agenten über sichtbaren Ausschnitt des Umgebungszustands
 - Aktionsbefehle von Agenten an die Umgebung
 - Authentifikation
 - ggf. Funktionalität zur Anbindung eines Beobachterwerkzeugs
- verschiedene Protokolle für:
 - **Umgebung** → **Agent**: geprägt von Zustandsinformationen
 - **Agent** → **Umgebung**: geprägt von Aktionsbefehlen
 - ggf. **Umgebung** → **Beobachterwerkzeug**: geprägt von erweiterten Zustandsinformationen
 - ggf. **Beobachterwerkzeug** → **Umgebung**: geprägt von „next step“-Befehlen
- Evaluation der *Knowledge Query and Manipulation Language (KQML)*

- sollen folgende Funktionen erfüllen:
 - Versand von Nachrichten durch Agenten über die Umgebung
 - zu spezifischem Agenten in „Flüster-Reichweite“
 - Multicast zu allen Agenten in Reichweite
 - Information der Agenten über sichtbaren Ausschnitt des Umgebungszustands
 - Aktionsbefehle von Agenten an die Umgebung
 - Authentifikation
 - ggf. Funktionalität zur Anbindung eines Beobachterwerkzeugs
- verschiedene Protokolle für:
 - **Umgebung** → **Agent**: geprägt von Zustandsinformationen
 - **Agent** → **Umgebung**: geprägt von Aktionsbefehlen
 - ggf. **Umgebung** → **Beobachterwerkzeug**: geprägt von erweiterten Zustandsinformationen
 - ggf. **Beobachterwerkzeug** → **Umgebung**: geprägt von „next step“-Befehlen
- Evaluation der *Knowledge Query and Manipulation Language (KQML)*

- plattformunabhängig in Java
- Umgebungs-Framework als Server-Software unabhängig von bestehenden Multiagentensystem-Frameworks
- grafischer benutzergesteuerter Test-Client
- einfache Testagenten unter Verwendung eines bestehenden Multiagentensystem-Frameworks
- ggf. Beobachterwerkzeug
- jegliche Kommunikation über TCP/IP
- Referenzimplementierung der agentenseitigen Schnittstelle in Java
- sehr wahrscheinlich: XML mit XML-Schemata für Spezifikationsformat und alle Kommunikationsprotokolle

Screenshot des aktuellen Prototyps



- 1 Einführung
 - Definitionen
 - Beispiel
- 2 Problemstellung
 - Überblick
 - Festlegung der Eigenschaften und Funktionen
 - Zustandsbeschreibung und -überführung
 - Spezifikationsformat
 - Kommunikationsprotokolle
 - Implementierung
- 3 Zusammenfassung

- Framework für gridbasierte virtuelle Welten/Umgebungen
- Konzeption
 - Funktionsumfang und Eigenschaften
 - formale Beschreibung von Zustand und Verhalten
 - Spezifikationsformat
 - Kommunikationsprotokolle
- Implementierung
 - Server-Software
 - grafischer Test-Client
 - einfache Testagenten
 - ggf. Beobachterwerkzeug



Jacques Ferber and Jean-Pierre Müller.

Influences and reaction: a model of situated multiagent systems.

In Proceedings of the Second International Conference on Multiagent Systems, pages 72–79, 1996.



Stuart J. Russell and Peter Norvig.

Artificial Intelligence: A Modern Approach.

Prentice Hall, Upper Saddle River, NJ, 3. edition, 2009.



Gerhard Weiss, editor.

Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence.

MIT Press, Cambridge, MA, 1999.