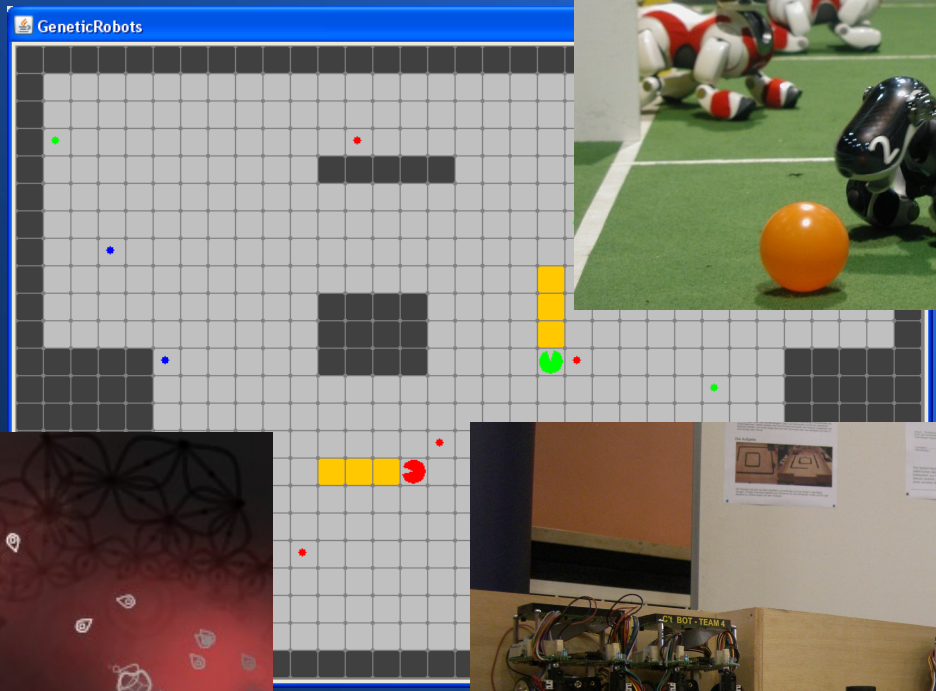
Domain Specific Languages für die Beschreibung von Agentenverhalten

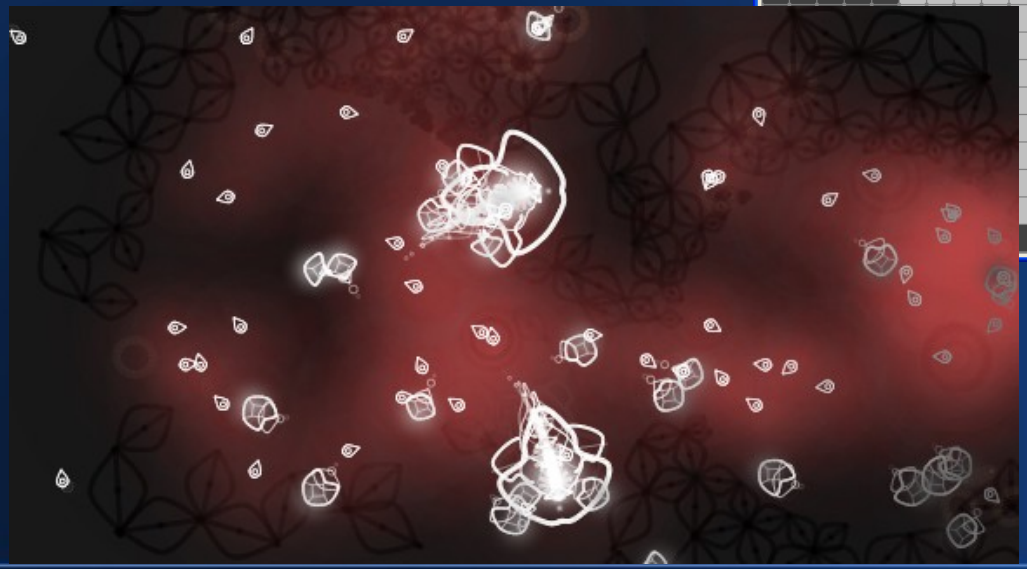
Agenda

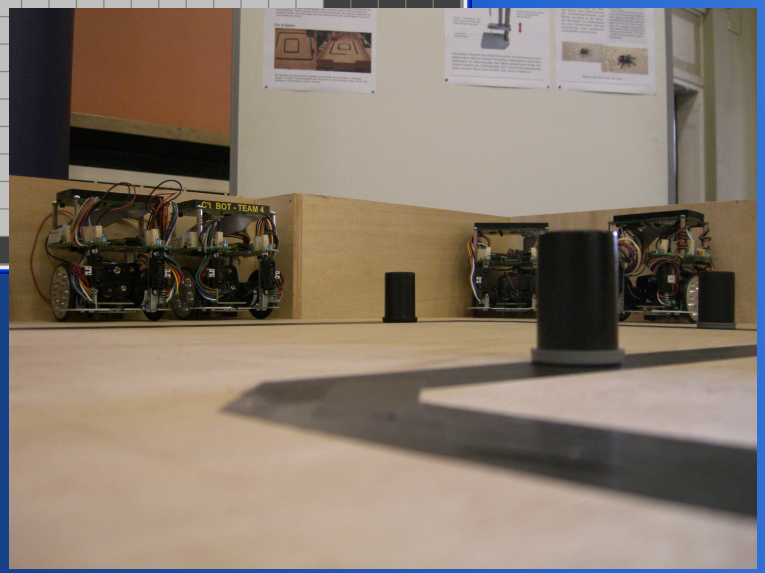
- Motivation
- Domain Specific Languages (DSL)
  - Was sind das?
  - Interne / Externe DSLs
- Intern
  - Jadex
  - Jack
- Extern
  - 2APL / 3APL
  - Jason [AgentSpeak(L)]
- Fazit

Motivation

- Erfahrung
  - Robocup
  - Projekt im Bachelor
  - Bachelorarbeit [10]
  - Master Projekt







Motivation

In den Projekten wurde immer in imperativen und Objektorientierten Sprachen das Verhalten modelliert. Zudem waren alle Systeme als reaktive Agenten ausgelegt. Z.T. aus Mangel an Ausdrucksmöglichkeiten.

Zentrale Frage:

Wie kann man Agentenverhalten besser beschreiben?

Domain Specific Languages

The basic idea of a domain specific language (DSL) is a computer language that's targeted to a particular kind of problem, rather than a general purpose language that's aimed at any kind of software problem. Domain specific languages have been talked about, and used for almost as long as computing has been done.

Fowler [1]

Domain Specific Languages

DSLs are very common in computing: examples include CSS, regular expressions, make, rake, ant, SQL, HQL, many bits of Rails, expectations in JMock, graphviz's dot language, FIT, strut's configuration file....

Fowler [1]

Extern

- Ist unabhängig von der Hostsprache und
- bildet in Syntax und Semantik eine eigenständige Sprache
- dadurch kann das Domänen spezifische Problem sehr genau abgebildet werden

Beispiel: Configuration Reader

Ein Beispiel von Martin Fowler [2] was eine DSL sein kann.

Zuerst haben wir ein Textfile:

```
#12345678901234567890123456789012345678901234567890123456789012345678901234567890
SVCLFOWLER          10101MS0120050313.......................
SVCLHOHPE           10201DX0320050315.......................
SVCLTWO            x10301MRP220050329.............................
USGE10301TWO           x50214..7050329.............................
```

```
Ein ObjektTyp      Die Felder des Objektes...
```

Beispiel: Configuration Reader

```
class Reader...
    public IList Process(StreamReader input) {
        IList result = new ArrayList();
        string line;
        while ((line = input.ReadLine()) != null)
            ProcessLine(line, result);
        return result;
    }
    private void ProcessLine(string line, IList result) {
        if (isBlank(line)) return;
        if (isComment(line)) return;
        string typeCode = GetTypeCode(line);
        IReaderStrategy strategy = (IReaderStrategy)_strategies[typeCode];
        if (null == strategy) throw new Exception("Unable to find strategy");
            result.Add(strategy.Process(line));
    }
    private static bool isComment(string line) { return line[0] == '#'; }
    private static bool isBlank(string line) { return line == ""; }
    private string GetTypeCode(string line) { return line.Substring(0,4); }

    IDictionary _strategies = new Hashtable();
    public void AddStrategy(IReaderStrategy arg) {
        _strategies[arg.Code] = arg;
    }
```

Beispiel: Configuration Reader

```
public void Configure(Reader target) {
    target.AddStrategy(ConfigureServiceCall());
    target.AddStrategy(ConfigureUsage());
}
private ReaderStrategy ConfigureServiceCall() {
    ReaderStrategy result = new ReaderStrategy("SVCL", typeof (ServiceCall));
    result.AddFieldExtractor(4, 18, "CustomerName");
    result.AddFieldExtractor(19, 23, "CustomerID");
    result.AddFieldExtractor(24, 27, "CallTypeCode");
    result.AddFieldExtractor(28, 35, "DateOfCallString");
    return result;
}
private ReaderStrategy ConfigureUsage() {
    ReaderStrategy result = new ReaderStrategy("USGE", typeof (Usage));
    result.AddFieldExtractor(4, 8, "CustomerID");
    result.AddFieldExtractor(9, 22, "CustomerName");
    result.AddFieldExtractor(30, 30, "Cycle");
    result.AddFieldExtractor(31, 36, "ReadDate");
    return result;
}
```

Intern

- Ist eingebettet in die Hostsprache
- Bildet ein Subset der Ausdrucksmöglichkeit der Hostsprache

Beispiele

- Method Chaining

```
computer()
    .processor()
      .cores(2)
      .i386()
    .disk()
      .size(150)
    .disk()
      .size(75)
      .speed(7200)
      .sata()
    .end();
```

- Function Sequence

```
computer();
    processor();
      cores(2);
      processorType(i386);
    disk();
      diskSize(150);
    disk();
      diskSize(75);
      diskSpeed(7200);
      diskInterface(SATA);
```

Vor- / Nachteile

Externe DSLs

- ***Vorteile***
  - Semantisch abgegrenzt
  - Unabhängig von Hostsprache
- ***Nachteile***
  - oft hat die IDE keine Kenntnis von der DSL, bzw. oft kommt DSL ohne IDE Integration
    - schwer zu Debuggen
    - Tools wie Refactoring nicht möglich zu benutzen

Interne DSLs

- ***Vorteile***
  - Volle Unterstützung der IDE, da es sich immer noch um die Hostsprache handelt
- ***Nachteile***
  - Semantisch nicht klar abgegrenzt
  - abhängig von Hostsprache

Die Suche nach:

- Die Sprache soll
  - die Domäne der deliberativen Agenten möglichst klar abbilden
  - integrierbar in bestehende Projekte sein (z.B. Java)
  - frei verfügbar, wenn möglich Open Source

- Alle folgenden Sprachen implementieren das BDI Paradigma, um deliberative Agenten zu beschreiben.

Interne Agentensprachen

- Jadex
- JACK

Jadex

- Entwickelt an der Universität Hamburg

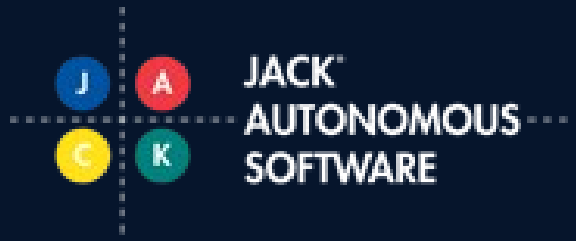
Agentenmodell	BDI
Sprache	Java / XML
Sprachkonzept	objektorientiert / imperativ
Agentenplattform	JADE
Lizenz	Open Source

Jadex

```
public class BroadcastVisitedPositionsPlan extends Plan {

	//-------- methods -------------

	/**
	 * The plan body.
	 */
	@Override
	public void body() {
		// fetch parameters:
		String teamID = (String) this.getBeliefbase().getBelief("teamID").getFact();

		// fetch parameter:
		MapPoint[] mps = (MapPoint[]) this.getParameterSet("visited_positions").getValues();

		// create message content:
		VisitedPositionsMessage msg_content = new VisitedPositionsMessage();
		msg_content.setMps(mps);

		AgentDescription[] agents = getTeamMembers(teamID);
		for (AgentDescription ad : agents) {                                    // send message
			if (! ad.getName().getLocalName().contains(this.getAgentName())) {     // don't send to self
				IMessageEvent mevent = createMessageEvent("visited_position_inform");
				mevent.setContent(msg_content);
				mevent.getParameterSet(SFipa.RECEIVERS).addValue(ad.getName());
				sendMessage(mevent);
			}
		}
		System.out.println(this.getAgentName() + " broadcasted visited positions");

	}
```

Jadex

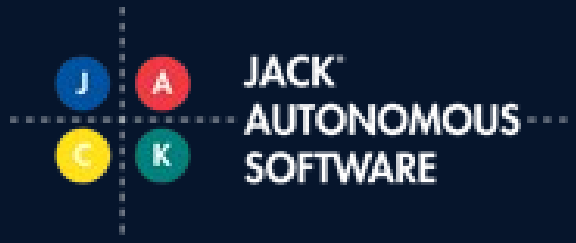
```
<agent xmlns="http://jadex.sourceforge.net/jadex"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://jadex.sourceforge.net/jadex
                          http://jadex.sourceforge.net/jadex-0.96.xsd"
      name="Cleaner"
      package="jadex.examples.cleanerworld.multi.cleaner">

      <beliefs>
            <!-- The known wastes. -->
            <beliefset name="wastes" class="Waste" />
      </beliefs>

      <goals>
            <!-- Broadcast waste object to team members. -->
            <achievegoal name="broadcast_waste" exported="true">
                  <parameter name="waste" class="Waste" />
            </achievegoal>
      </goals>

      <plans>
            <!-- Broadcast waste-object to team members. -->
            <plan name="broadcast_waste">
                  <parameter name="waste" class="Waste">
                        <goalmapping ref="broadcast_waste.waste" />
                  </parameter>
                  <body class="BroadcastWastePlan"></body>
                  <trigger>
                        <goal ref="broadcast_waste"/>
                  </trigger>
            </plan>
      </plans>
</agent>
```



JACK

- Ein Produkt der *Autonomous Decision-Making Software*

Agentenmodell	BDI
Sprache	Java (Angepasste Syntax)
Sprachkonzept	objektorientiert / imperativ
Agentenplattform	proprietär?
Lizenz	kommerziell



Jack

```
agent ExampleAgent extends Agent {
      // #-statements
      // data members

      ExampleAgent (String name)
       {
            super(name);
            ...
      }
}
```

```
plan MovementResponse extends Plan {

  #handles event RobotMoveEvent moveResponse;
  #uses agent implementing RobotInterface robot;

  static boolean relevant (RobotMoveQEvent ev) {
    return (ev.ID == RobotMoveQEvent.REPLY_SAFE ||
            ev.ID == RobotMoveQEvent.REPLY_DEAD);
  }

  Body() {
    if (moveResponse.ID==REPLAY_SAFE)
    {
      System.err.println("Robot Safe to Move");
      robot.updatePosition(moveResponse.Lane,
            moveResponse.Displacement);
    } else {
      // robot didn't make it
      System.err.println("Robot is Dead");
      robot.die();
    }
  }
}
```

Externe Agentensprachen

- 3APL
- 2APL
- Jason [AgentSpeak(L)]

3APL – An Abstract Agent Programming Language

- Entwickelt an der Utrecht University (Niederlande)

Agentenmodell	BDI
Sprache	3APL
Sprachkonzept	imperativ / logisch (Prolog Kern)
Agentenplattform	Eigenentwicklung orientiert sich an FIPA
Implementierung	Java / Haskel
Lizenz	k.A.

3APL – An Abstract Agent Programming Language

```
PROGRAM "cleaning"
CAPABILITIES{
   { pos(P) }                  Goto(R)          { NOT pos(P) , pos(R) },
   { pos(P) AND dirty(R) }     Vacuum(R)        { NOT dirty(R) },
   { pos(P1) AND box(P1)}      Movebox(P1,P2)   { NOT pos(P1), NOT box(P1), pos(P2), box(P2)},
   {TRUE}                      IsClean()        {clean()},
   {TRUE}                      Transported()    {transport()}
}
BELIEFBASE{
   dirty(room1).
   dest(room1).
   box(room2).
   pos(room3).
}
GOALBASE{ clean(), transport() }
PLANBASE{ }
PG-RULES{
   clean() <- dirty(Room) |
   { Goto(Room);
     Vacuum(Room);
     if not dirty(R) then IsClean()
   },
   transport() <- box(Room) AND dest(Dest) |
   { Goto(Room);
     Movebox(Room,Dest);
     if box(Dest) then Transported()
   }
}
PR-RULES{}
```

A Practical Agent Programming Language

- Entwickelt an der Utrecht University (Niederlande)
- Nachfolger von 3APL

Agentenmodell	BDI
Sprache	2APL
Sprachkonzept	imperativ / logisch (Prolog Kern)
Agentenplattform	JADE
Implementierung	Java
Lizenz	k.A.

A Practical Agent Programming Language

```
Include: person.2apl

Goals:
  search( blockWorld )

Plans:
  @blockworld( enter( 8, 8, red ), L )

PG-rules:
  search( blockWorld ) <- true |
  {
    B(is( X, int( random( 15 ) ) ));
    B(is( Y, int( random( 15 ) ) ));
    goto( X, Y );
    @blockworld( senseBombs(), BOMBS );
    if B( BOMBS = [ [default,X1,Y1] | REST ] ) then
    { send( harry, inform, La, On, bombAt( X1, Y1 ) ) }
  }
```

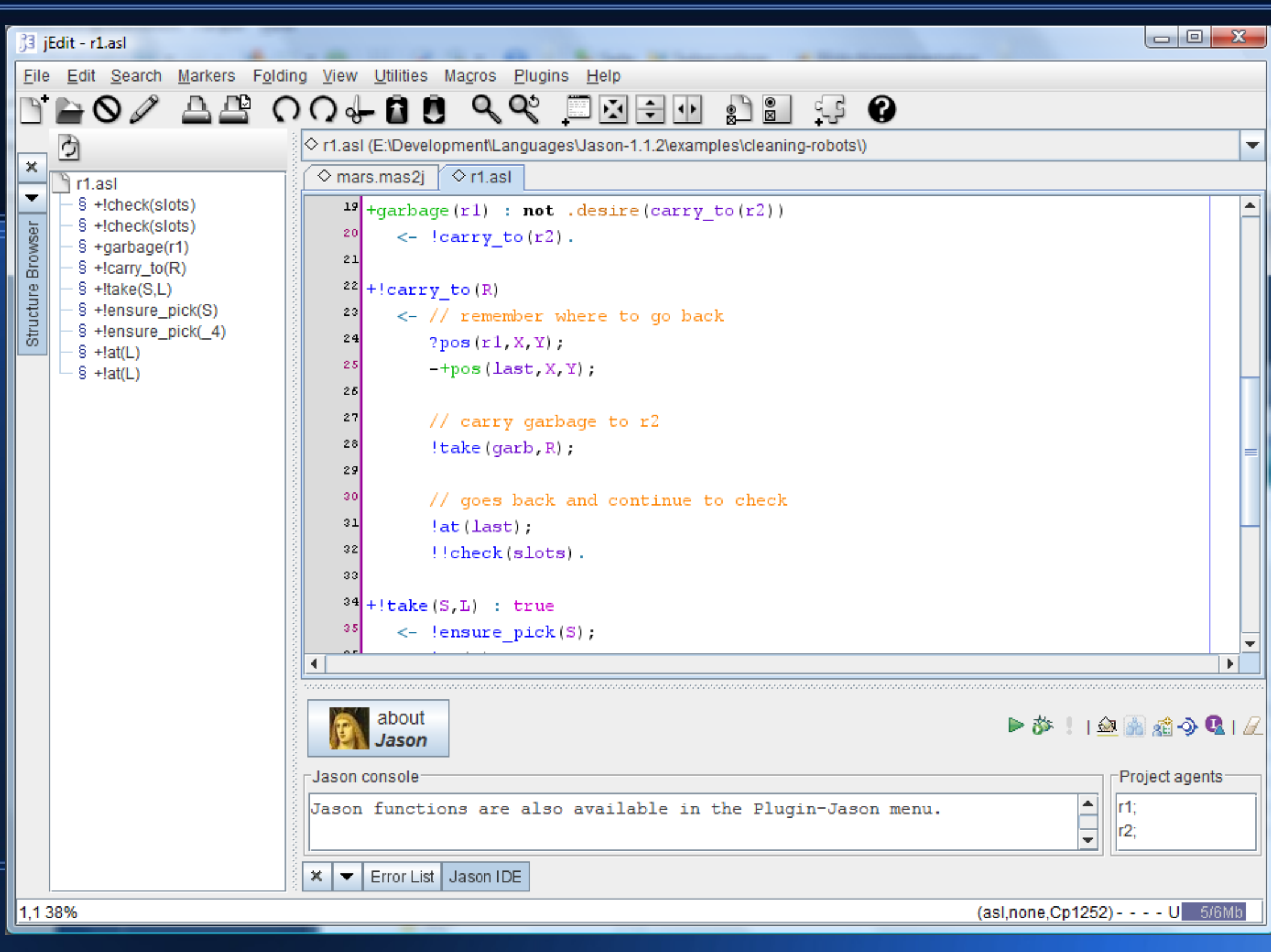
eine AgentSpeak(L) Implementierung

- Entwickelt von Jomi F. Hübner und Rafael H. Bordini

Agentenmodell	BDI
Sprache	AgentSpeak(L)
Sprachkonzept	logisch
Agentenplattform	SACI / JADE
Implementierung	Java
Lizenz	OpenSource

```
// mars robot 1

/* Initial beliefs */

at(P) :- pos(P,X,Y) & pos(r1,X,Y).

/* Initial goal */

!check(slots).

/* Plans */

+!check(slots) : not garbage(r1)
   <- next(slot);
      !!check(slots).
+!check(slots).


+garbage(r1) : not .desire(carry_to(r2))
   <- !carry_to(r2).
```

```
+!carry_to(R)
   <- // remember where to go back
      ?pos(r1,X,Y);
      -+pos(last,X,Y);

      // carry garbage to r2
      !take(garb,R);

      // goes back and continue to check
      !at(last);
      !!check(slots).

+!take(S,L) : true
   <- !ensure_pick(S);
      !at(L);
      drop(S).

+!ensure_pick(S) : garbage(r1)
   <- pick(garb);
      !ensure_pick(S).
+!ensure_pick(_).

+!at(L) : at(L).
+!at(L) <- ?pos(L,X,Y);
           move_towards(X,Y);
           !at(L).
```


jEdit - r1.asl
File Edit Search Markers Folding View Utilities Macros Plugins Help
r1.asl (E:\Development\Languages\Jason-1.1.2\examples\cleaning-robots\)
mars.mas2j
r1.asl
Structure Browser
r1.asl
§ +!check(slots)
§ +!check(slots)
§ +garbage(r1)
§ +!carry_to(R)
§ +!take(S,L)
§ +!ensure_pick(S)
§ +!ensure_pick(_4)
§ +!at(L)
§ +!at(L)
19 +garbage(r1) : not .desire(carry_to(r2))
20 <- !carry_to(r2).
21
22 +!carry_to(R)
23 <- // remember where to go back
24 ?pos(r1,X,Y);
25 -+pos(last,X,Y);
26
27 // carry garbage to r2
28 !take(garb,R);
29
30 // goes back and continue to check
31 !at(last);
32 !!check(slots).
33
34 +!take(S,L) : true
35 <- !ensure_pick(S);
about Jason
Jason console
Jason functions are also available in the Plugin-Jason menu.
Project agents
r1;
r2;
Error List
Jason IDE
1,1 38%
(asl,none,Cp1252) - - - - U
5/6Mb

Fazit

- es gibt einige wenige aktiv gepflegte Sprachen
- Fast alle kommen aus Universitäten
- Jason ist die einzige, die meine Kriterien erfüllt
- Verwendung der externen Sprachen in großen bzw. kommerziellen Projekten unklar oder gar nicht

Quellen

- [1]: Martin Fowler – Domain Specific Languages – http://www.martinfowler.com/bliki/DomainSpecificLanguage.html
- [2]: Martin Fowler – Language Workbenches – http://martinfowler.com/articles/languageWorkbench.html
- [3]: Auf dem Weg zu idealen Programmierwerkzeugen – Informatik Spektrum Band 31 Heft 6 – S. 580
- [4]: Jadex – http://vsis-www.informatik.uni-hamburg.de/projects/jadex/
- [5]: JACK – http://www.agent-software.com/
- [6]: 2APL – Utrecht University (Niederlande) – http://www.cs.uu.nl/2apl/
- [7]: 3APL – Utrecht University (Niederlande) – http://www.cs.uu.nl/3apl/

Quelle

- [8]: Jason – Jomi F. Hübner, Rafael H. Bordini – http://jason.sourceforge.net/JasonWebSite/Jason%20Home.php
- [9]: SACI – Simple Agent Communication Infrastructure – Jomi Fred Hübner, Jaime Simão Sichman – Universidade de São Paulo – http://www.lti.pcs.usp.br/saci/