COINED
Collaborative Examination of Object Interaction

Till Schümmer & Petra Kösters
till.schuemmer@fernuni-hagen.de

Cooperative Systems
FernUniversität in Hagen
Germany

Goals

- Understanding objects
  - Domain model
  - Behavior
  - Interaction
- Understanding classes
  - Interfaces & inheritance
- Understanding test cases

Approach

- Users direct objects to simulate responses to messages
- Dynamics are visualized in a shared diagram editor
- Objects can receive any message
- Unknown messages help to refine the classes
- Message traces are the starting points for the generation of unit tests and class stubs

The COINED Environment

Collaboration Space

Users can meet in an object space and start to direct instances of a class. All users share the same view on the object space.

Test-Case Generator

The history keeps track of message sends and the created return values. These are used to generate automatic Unit tests.

To ease the implementation, stubs of the used classes can also be created.

Object Repository

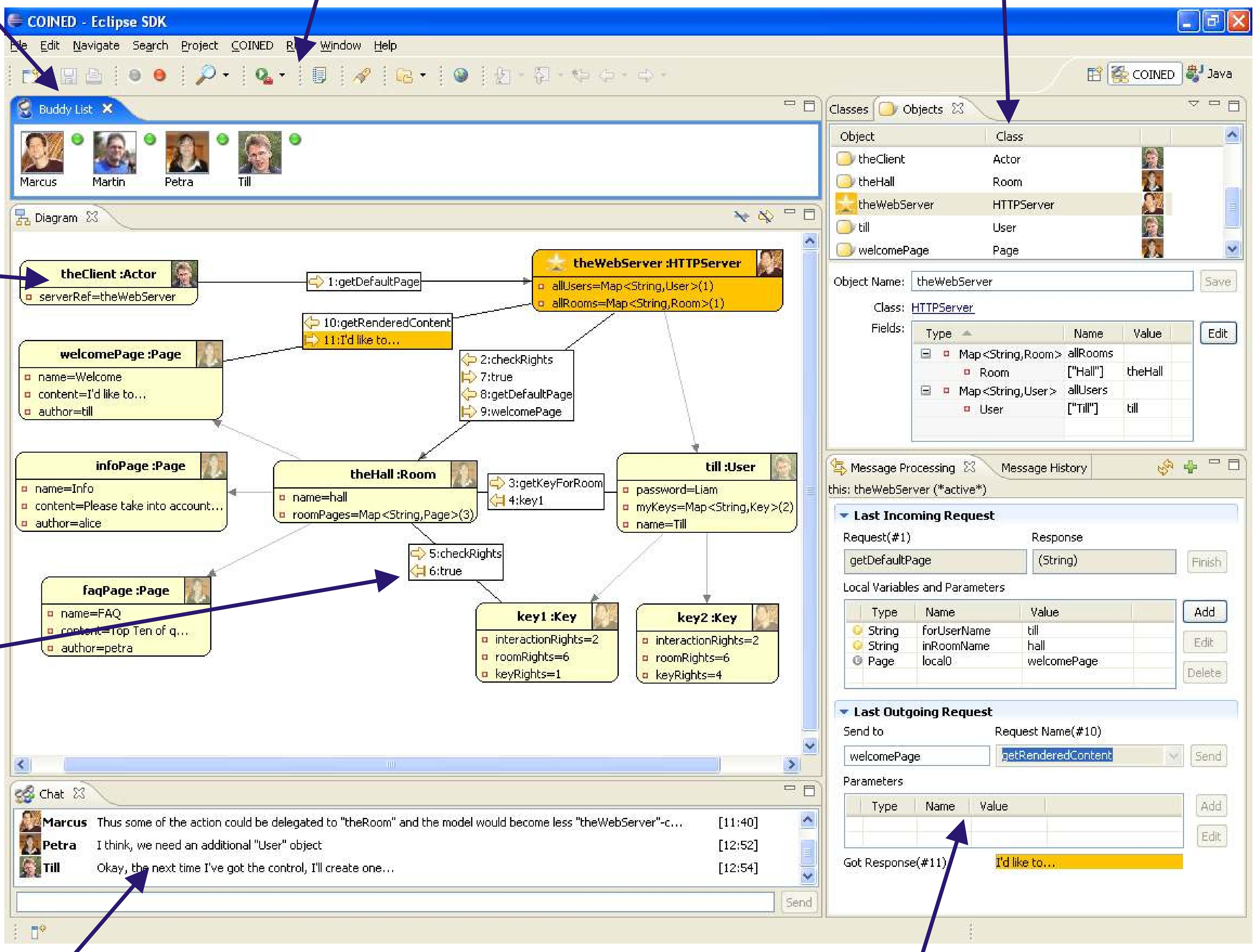
The directing user keeps track of all her objects and can manipulate them when active. Foreign objects are shown but non-directing users can only send messages to the object.

Class Repository

All users can access all classes to investigate interfaces or modify methods. They can become a class director to create new instances and control the instances' behavior.

Object Representation

Each object shows the director who takes care of the instance. Active objects are highlighted. Objects that are part of the current execution stack are shown with a coloured user icon. Other objects use a faded icon.



Chat

Marcus	Thus some of the action could be delegated to "theRoom" and the model would become less "theWebServer"-c...	[11:40]
Petra	I think, we need an additional "User" object	[12:52]
Till	Okay, the next time I've got the control, I'll create one...	[12:54]

Objects

Object	Class
theClient	Actor
theHall	Room
theWebServer	HTTPServer
till	User
welcomePage	Page

Object Name: theWebServer
Class: HTTPServer
Fields:

Type	Name	Value
Map<String,Room>	allRooms	
Room	["Hall"]	theHall
Map<String,User>	allUsers	
User	["Till"]	till

Message Processing
this: theWebServer (*active*)

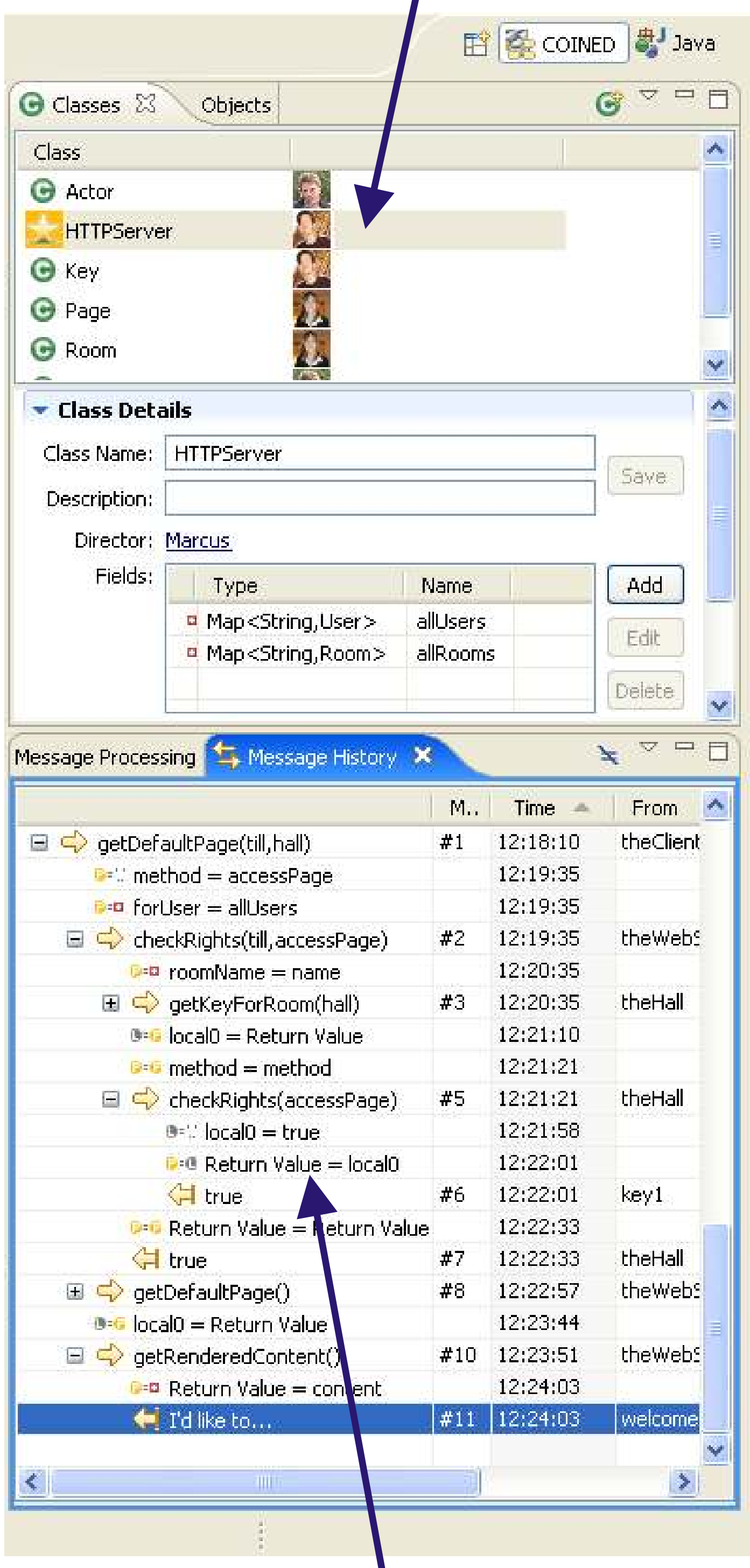
Last Incoming Request
Request(#1): getDefaultPage
Response: (String)
Local Variables and Parameters

Type	Name	Value
String	forUserName	till
String	inRoomName	hall
Page	local0	welcomePage

Last Outgoing Request
Send to: welcomePage
Request Name(#10): getRenderedContent
Parameters

Type	Name	Value

Got Response(#11): I'd like to...



Class
Actor
HTTPServer
Key
Page
Room

Class Name: HTTPServer
Description:
Director: Marcus
Fields:

Type	Name
Map<String,User>	allUsers
Map<String,Room>	allRooms

Message History

	M..	Time	From
getDefaultPage(till,hall)	#1	12:18:10	theClient
method = accessPage		12:19:35	
forUser = allUsers		12:19:35	
checkRights(till,accessPage)	#2	12:19:35	theWebS
roomName = name		12:20:35	
getKeyForRoom(hall)	#3	12:20:35	theHall
local0 = Return Value		12:21:10	
method = method		12:21:21	
checkRights(accessPage)	#5	12:21:21	theHall
local0 = true		12:21:58	
Return Value = local0		12:22:01	
true	#6	12:22:01	key1
Return Value = Return Value		12:22:33	
true	#7	12:22:33	theHall
getDefaultPage()	#8	12:22:57	theWebS
local0 = Return Value		12:23:44	
getRenderedContent()	#10	12:23:51	theWebS
Return Value = content		12:24:03	
I'd like to...	#11	12:24:03	welcome

Message Flow

Messages and answer values of the current message history decorate object relations. Users can control the depth of the message history shown in the diagram.

Meta Communication

Users discuss the current execution flow using a persistent chat. The chat transcript will be part of the generated class stubs and the unit tests.

Processing Area

When objects receive a message, it is displayed in the processing area. The director has the task of generating a response to the message. This can mean that local values are stored or manipulated, attributes of the receiver are updated, or new messages are sent to objects known or created by the receiver. Values can be dragged from the processing area to attributes of the active object. When a message is sent, control is passed on to the director of the receiving object.

Message History

Messages are stored for later reference. Users can inspect the message flow to reflect on the interaction that took place in the simulation.

Current Status and Evaluation

- COINED is an Eclipse plugin that integrates with the Eclipse Java development environment
- Currently planning and looking for classroom application in order to conduct user studies
- Tools will be combined with other plugins, e.g., for supporting distributed planning games
- To request the research prototype contact Till Schümmer (till.schuemmer@fernuni-hagen.de)