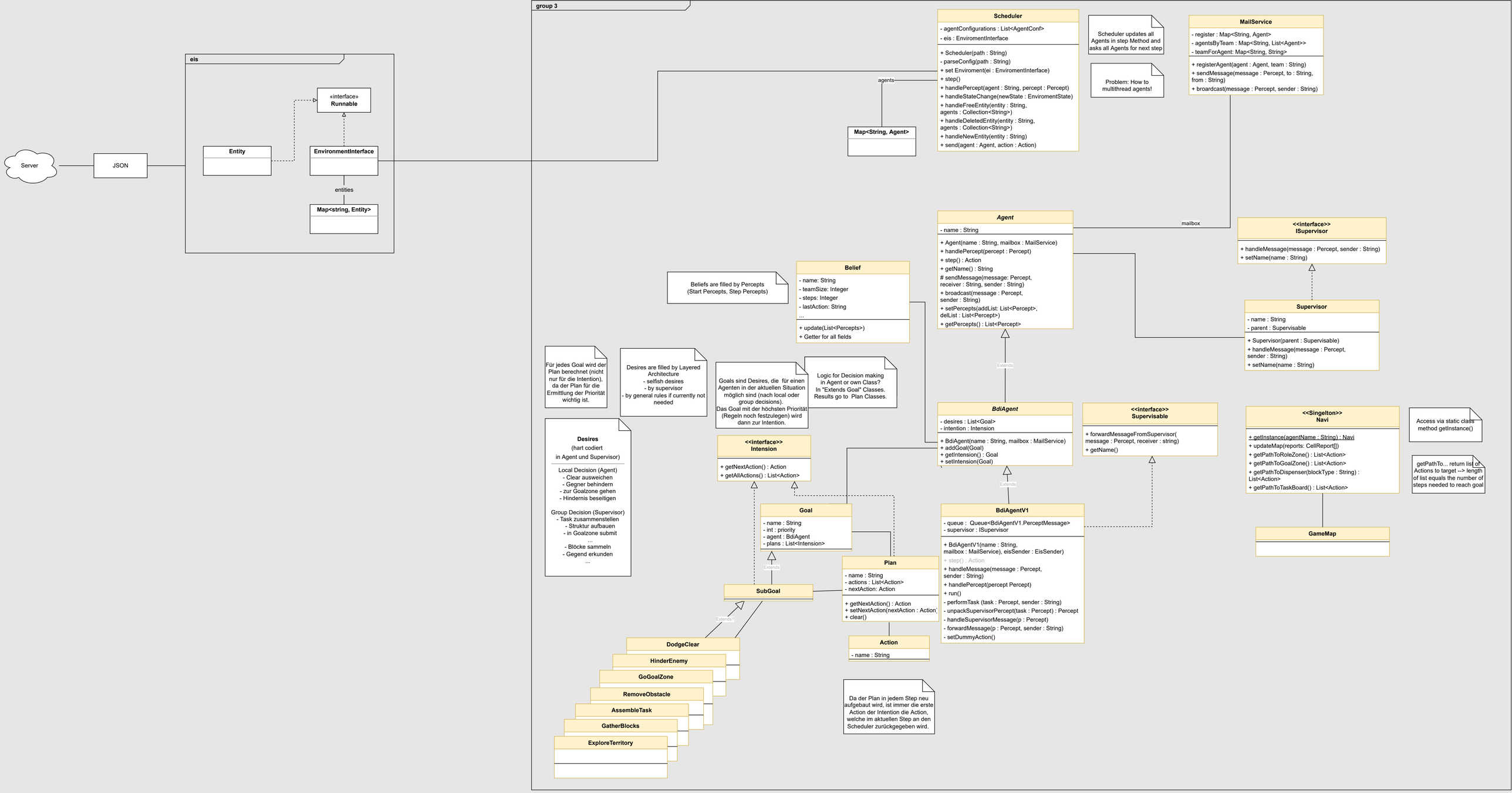
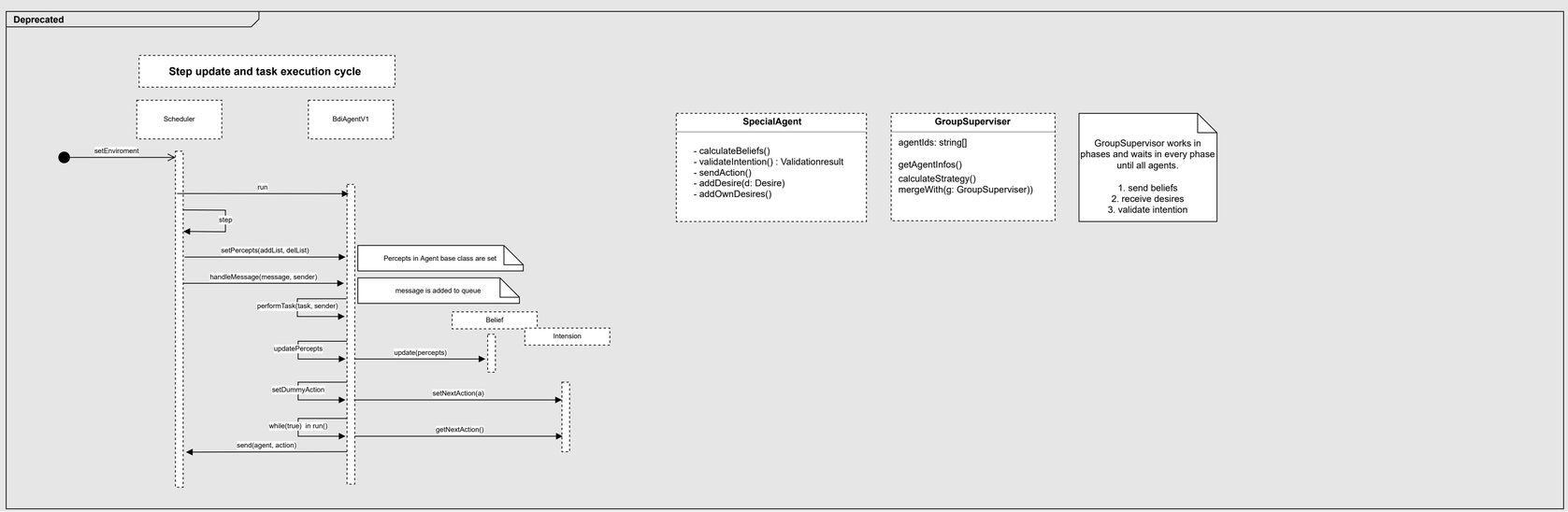

group 3
eis
Server
JSON
Entity
«interface»
Runnable
EnvironmentInterface
entities
Map<string, Entity>
agents
Map<String, Agent>
Scheduler
- agentConfigurations : List<AgentConf>
- eis : EnviromentInterface
+ Scheduler(path : String)
- parseConfig(path : String)
+ set Enviroment(ei : EnviromentInterface)
+ step()
+ handlePercept(agent : String, percept : Percept)
+ handleStateChange(newState : EnviromentState)
+ handleFreeEntity(entity : String, agents : Collection<String>)
+ handleDeletedEntity(entity : String, agents : Collection<String>)
+ handleNewEntity(entity : String)
+ send(agent : Agent, action : Action)
Scheduler updates all Agents in step Method and asks all Agents for next step
Problem: How to multithread agents!
MailService
- register : Map<String, Agent>
- agentsByTeam : Map<String, List<Agent>>
- teamForAgent: Map<String, String>
+ registerAgent(agent : Agent, team : String)
+ sendMessage(message : Percept, to : String, from : String)
+ broadcast(message : Percept, sender : String)
mailbox
Agent
- name : String
+ Agent(name : String, mailbox : MailService)
+ handlePercept(percept : Percept)
+ step() : Action
+ getName() : String
# sendMessage(message: Percept, receiver : String, sender : String)
+ broadcast(message : Percept, sender : String)
+ setPercepts(addList: List<Percept>, delList : List<Percept>)
+ getPercepts() : List<Percept>
<<interface>>
ISupervisor
+ handleMessage(message : Percept, sender : String)
+ setName(name : String)
Supervisor
- name : String
- parent : Supervisable
+ Supervisor(parent : Supervisable)
+ handleMessage(message : Percept, sender : String)
+ setName(name : String)
Beliefs are filled by Percepts (Start Percepts, Step Percepts)
Belief
- name: String
- teamSize: Integer
- steps: Integer
- lastAction: String
...
+ update(List<Percepts>)
+ Getter for all fields
Für jedes Goal wird der Plan berechnet (nicht nur für die Intention), da der Plan für die Ermittlung der Priorität wichtig ist.
Desires are filled by Layered Architecture
- selfish desires
- by supervisor
- by general rules if currently not needed
Goals sind Desires, die für einen Agenten in der aktuellen Situation möglich sind (nach local oder group decisions).
Das Goal mit der höchsten Priorität (Regeln noch festzulegen) wird dann zur Intention.
Logic for Decision making in Agent or own Class?
In "Extends Goal" Classes.
Results go to Plan Classes.
Extends
BdiAgent
- desires : List<Goal>
- intention : Intension
+ BdiAgent(name : String, mailbox : MailService)
+ addGoal(Goal)
+ getIntension() : Goal
+ setIntension(Goal)
<<interface>>
Supervisable
+ forwardMessageFromSupervisor(message : Percept, receiver : string)
+ getName()
<<Singelton>>
Navi
+ getInstance(agentName : String) : Navi
+ updateMap(reports: CellReport[])
+ getPathToRoleZone() : List<Action>
+ getPathToGoalZone() : List<Action>
+ getPathToDispenser(blockType : String) : List<Action>
+ getPathToTaskBoard() : List<Action>
Access via static class method getInstance()
getPathTo... return list of Actions to target --> length of list equals the number of steps needed to reach goal
Desires
(hart codiert
in Agent und Supervisor)
Local Decision (Agent)
- Clear ausweichen
- Gegner behindern
- zur Goalzone gehen
- Hindernis beseitigen
Group Decision (Supervisor)
- Task zusammenstellen
- Struktur aufbauen
- in Goalzone submit
...
- Blöcke sammeln
- Gegend erkunden
...
<<interface>>
Intension
+ getNextAction() : Action
+ getAllActions() : List<Action>
Goal
- name : String
- int : priority
- agent : BdiAgent
- plans : List<Intension>
Extends
BdiAgentV1
- queue : Queue<BdiAgentV1.PerceptMessage>
- supervisor : ISupervisor
+ BdiAgentV1(name : String, mailbox : MailService), eisSender : EisSender)
+ step() : Action
+ handleMessage(message : Percept, sender : String)
+ handlePercept(percept Percept)
+ run()
- performTask (task : Percept, sender : String)
- unpackSupervisorPercept(task : Percept) : Percept
- handleSupervisorMessage(p : Percept)
- forwardMessage(p : Percept, sender : String)
- setDummyAction()
GameMap
Plan
- name : String
- actions : List<Action>
- nextAction: Action
+ getNextAction() : Action
+ setNextAction(nextAction : Action)
+ clear()
SubGoal
Extends
Action
- name : String
Da der Plan in jedem Step neu aufgebaut wird, ist immer die erste Action der Intention die Action, welche im aktuellen Step an den Scheduler zurückgegeben wird.
DodgeClear
HinderEnemy
GoGoalZone
RemoveObstacle
AssembleTask
GatherBlocks
ExploreTerritory


Deprecated
Step update and task execution cycle
Scheduler
BdiAgentV1
setEnviroment
run
step
setPercepts(addList, delList)
Percepts in Agent base class are set
handleMessage(message, sender)
message is added to queue
performTask(task, sender)
Belief
Intension
updatePercepts
update(percepts)
setDummyAction
setNextAction(a)
while(true) in run()
getNextAction()
send(agent, action)
SpecialAgent
- calculateBeliefs()
- validateIntention() : Validationresult
- sendAction()
- addDesire(d: Desire)
- addOwnDesires()
GroupSuperviser
agentIds: string[]
getAgentInfos()
calculateStrategy()
mergeWith(g: GroupSuperviser))
GroupSupervisor works in phases and waits in every phase until all agents.
1. send beliefs
2. receive desires
3. validate intention