Extensions of the BDI Architecture as Reusable Solutions to Ease the Development of Sophisticated BDI Agents

Ingrid Nunes
Universidade Federal do Rio Grande do Sul (UFRGS)
Porto Alegre, Brazil

WESAAC 2019 – May 4, 2019 – Florianópolis, BR

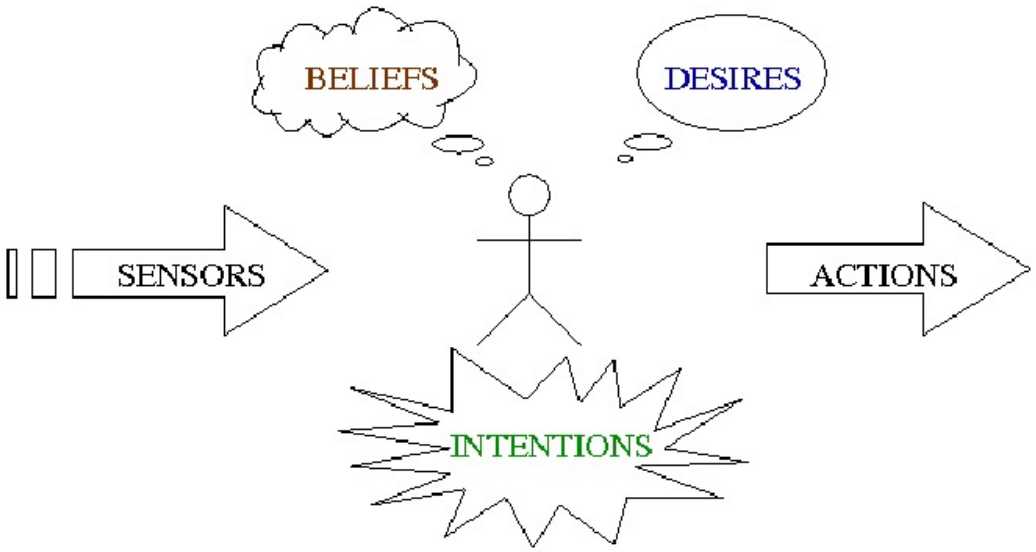
INSTITUTO DE INFORMÁTICA UFRGS

Prosoft

UFRGS UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

BDI Architecture

- Belief-Desire-Intention (BDI)
  - Model of agency with roots in practical reasoning and intentional systems
    - Set of concepts for thinking about and building agents
  - Behavior arises due to the agent
    - Committing to some of its **desires**
    - Selecting **actions** that achieve its intentions given its **beliefs**

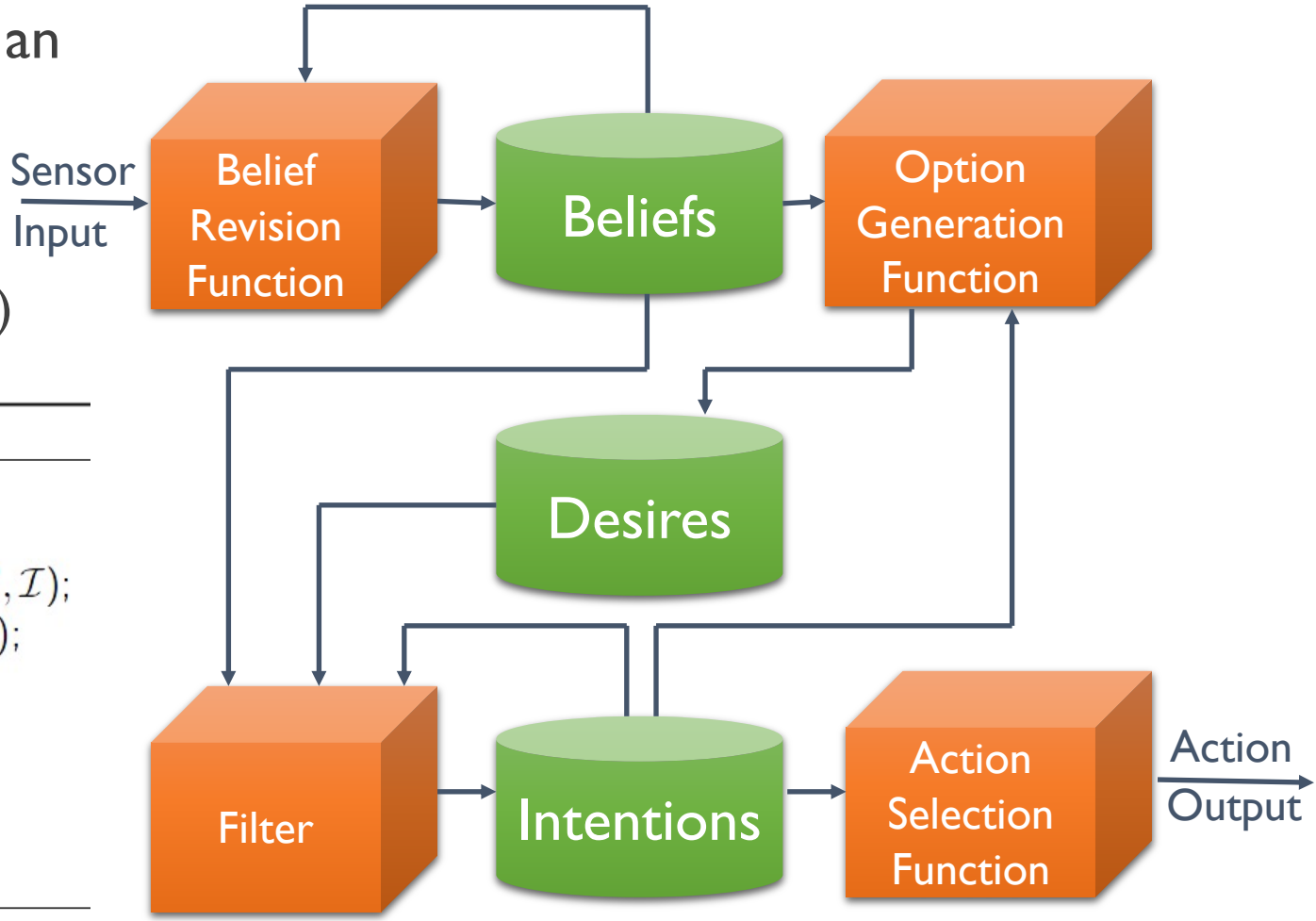


Abstract Rational Architecture

- Rao and Georgeff (1995) proposed an abstract interpreter for agents
- Key: the notion of events
  - the inputs of the system
  - both externals (from the environment) and internal (within the system)

Algorithm 1 BDI Rational Agent Interpreter

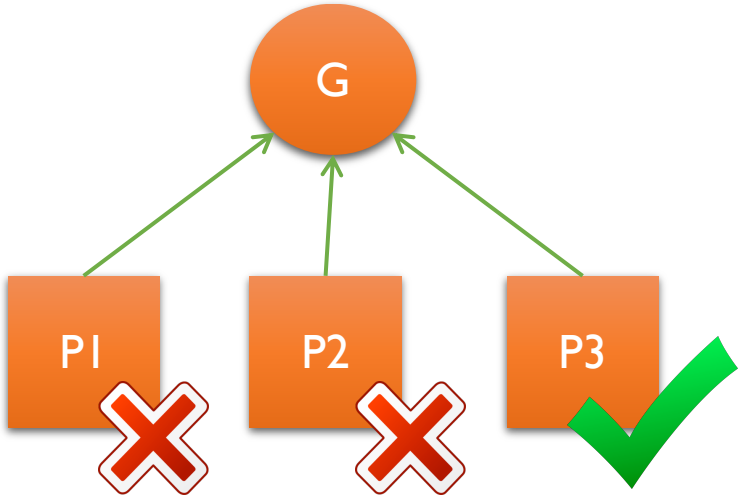
```
1: initialize_state()
2: while ¬quit do
3:    options ⇐ option_generator(event_queue, B, G, I);
4:    selected_options ⇐ deliberate(options, B, G, I);
5:    update_intentions(selected_options, I);
6:    execute(I);
7:    drop_successful_attitudes(B, G, I);
8:    drop_impossible_attitudes(B, G, I);
9: end while
```



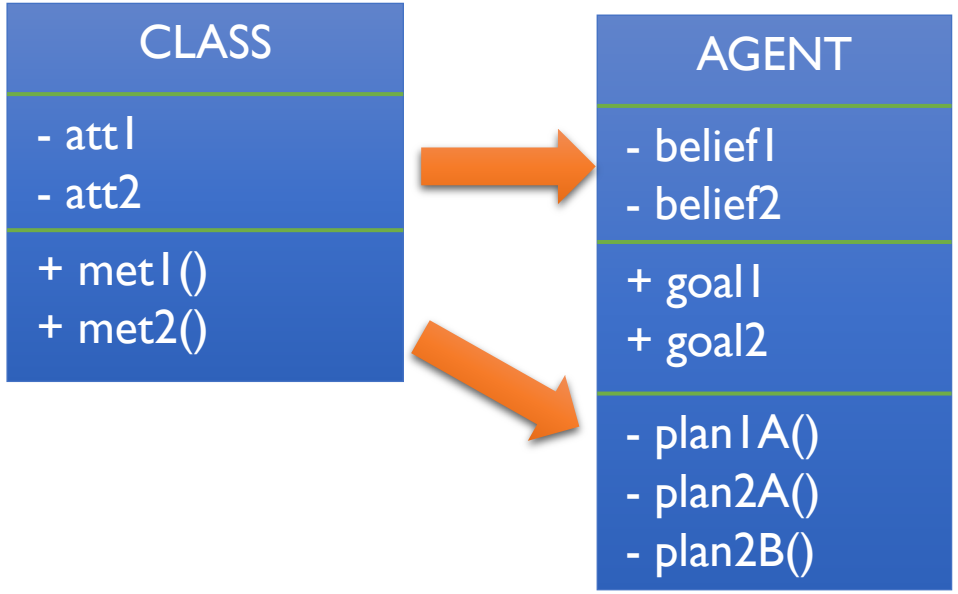
Rao, A. S. & Georgeff, M. P. (1995), BDI-agents: from theory to practice, in Proceedings of the First Intl. Conference on Multiagent Systems (ICMAS 1995), San Francisco, USA.

BDI Architecture

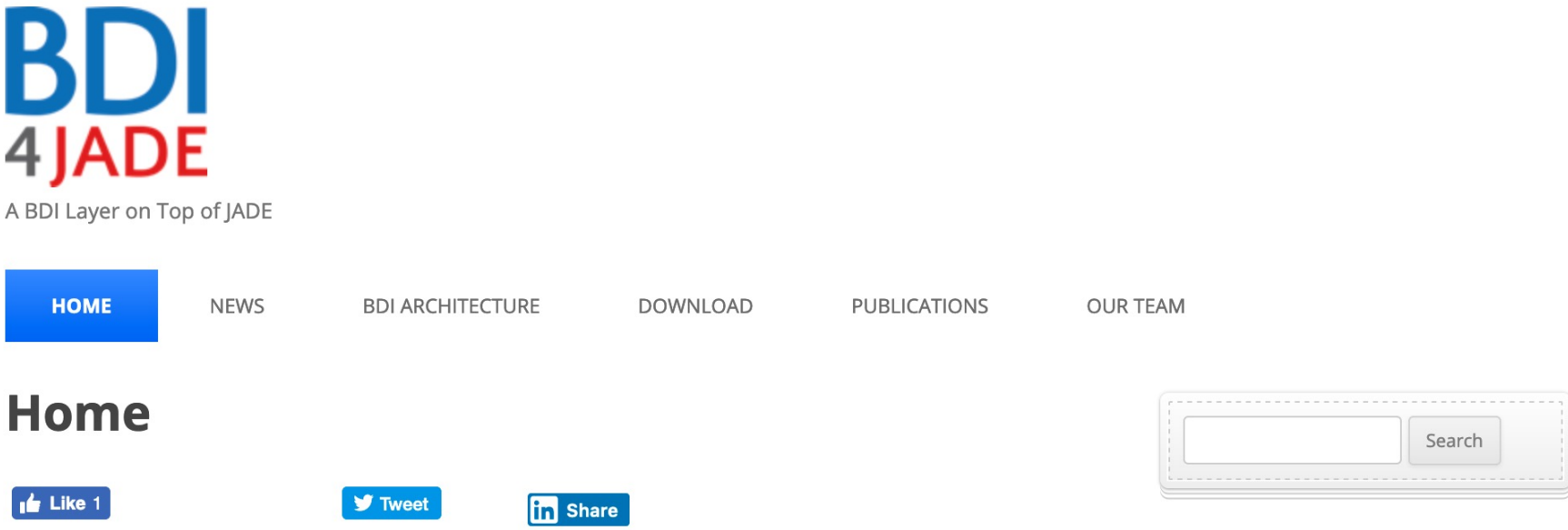
Goal vs. Plans



Objects vs. BDI Agents



BDI4JADE


BDI
4JADE
A BDI Layer on Top of JADE
HOME
NEWS
BDI ARCHITECTURE
DOWNLOAD
PUBLICATIONS
OUR TEAM
Home
Search
Like 1
Tweet
Share
BDI4JADE is an agent platform that implements the BDI (belief-desire-intention) architecture. It consists of a BDI layer implemented on top of JADE.
BDI4JADE leverages all the features provided by JADE and reuses it as much as possible. Other highlights of our JADE extension, besides providing BDI abstractions and the reasoning cycle, include:
DOWNLOAD
Download the latest version of BDI4JADE here. Its API can accessed online here.
Use of Capabilities – agents aggregate a set of capabilities, which are a collection of be-

BDI4JADE

```
public class PingPongCapability extends Capability {

    @GoalOwner(capability = PingPongCapability.class)
    public static class PingGoal implements Goal {
        private static final long serialVersionUID = -7733145369836002329L;
    }

    private static final long serialVersionUID = -4800805796961540570L;

    @Belief
    String neighbour;

    @bdi4jade.annotation.Plan
    private Plan pingPlan = new DefaultPlan(PingGoal.class, PingPlanBody.class);

    @Belief
    Integer pingTimes;

    @bdi4jade.annotation.Plan
    private Plan pongPlan = new DefaultPlan(MessageTemplate.MatchContent(PingPlanBody.MSG_CONTENT), PongPlanBody.class);

    public PingPongCapability(String neighbour, int pingTimes) {
        this.neighbour = neighbour;
        this.pingTimes = pingTimes;
    }

}
```

BDI4JADE

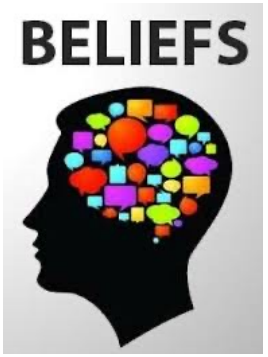
```
public class PongPlanBody extends AbstractPlanBody {

    private static final Log log = LogFactory.getLog(PongPlanBody.class);
    public static final String MSG_CONTENT = "pong";
    private static final long serialVersionUID = -3352874506241004611L;

    private ACLMessage pingMsg;

    @Override
    public void action() {
        log.info("Ping received from agent " + pingMsg.getSender().getName()
                + "!");
        ACLMessage reply = pingMsg.createReply();
        reply.setContent(MSG_CONTENT);
        this.myAgent.send(reply);
        log.info("Pong sent to agent " + pingMsg.getSender().getName() + "!");
        setEndState(EndState.SUCCESSFUL);
    }

    @Parameter(direction = Direction.IN)
    public void setMessage(ACLMessage pingMsg) {
        this.pingMsg = pingMsg;
    }

}
```

BDI4JADE

```
public PingPongAction() {
    super.putValue(Action.NAME, "Ping Pong Agents");
    this.agent1 = new SingleCapabilityAgent(new PingPongCapability(AGENT_2, 2));
    this.agent2 = new SingleCapabilityAgent(new PingPongCapability(AGENT_1, 1));
}

@Override
public void actionPerformed(ActionEvent e) {
    this.agent1.addGoal(new PingPongCapability.PingGoal());
    this.agent2.addGoal(new PingPongCapability.PingGoal());
}
```

BDI Architecture

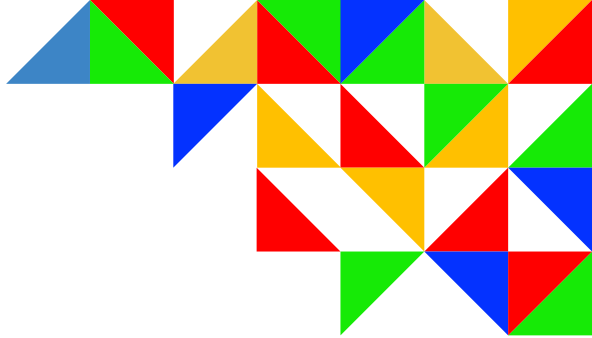
- **Flexible** architecture to implement cognitive agents
- Separation of **Goals** and **Plans**
- **Customisation Functions**
  - Key issue



- Existing techniques
  - Direct use by **mainstream** software **developers** is still **problematic**
    - **Expert knowledge** is still required to develop BDI agents
    - Real **barrier** to the **large-scale adoption** of this kind of agent technology

Agenda

1. Network Function Virtualisation Orchestration with BDI Agents
2. Extensions to the BDI Architecture
   A. Plan Selection based on Multi-attribute Utility Theory (MAUT)
   B. Learning-based Plan Selection
3. Automated Management of Remediation Actions

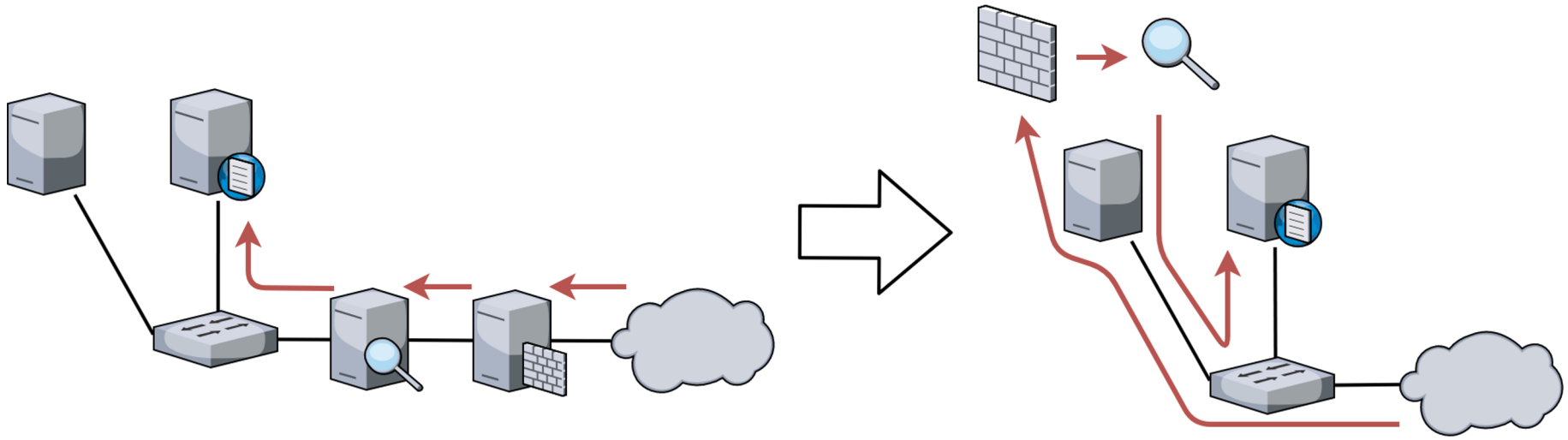


NFV Orchestration with BDI Agents

Extensions of the BDI Architecture as Reusable Solutions to Ease the Development of Sophisticated BDI Agents

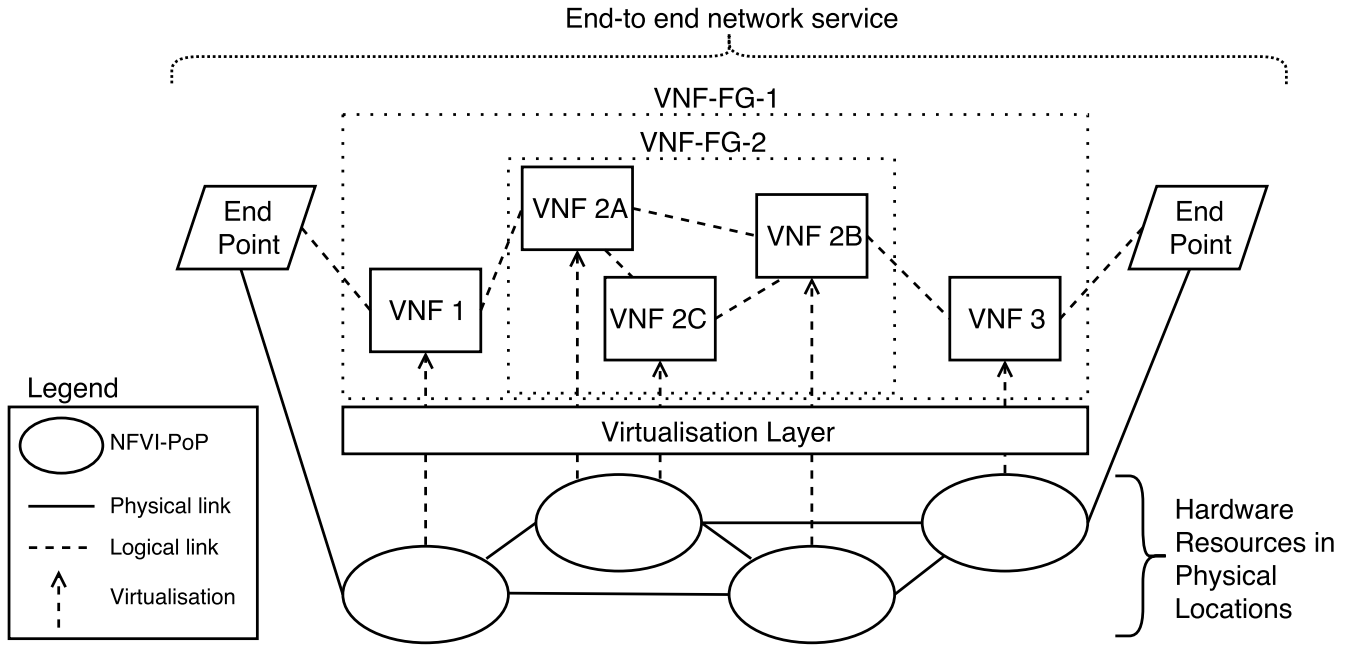
NFV Orchestration with BDI Agents

- Network Functions Virtualisation (NFV)
  - Decouples network services from physical devices (middleboxes) through virtualization
  - Middleboxes can be virtualised into cheap and easily deployable virtual machines



NFV Orchestration with BDI Agents

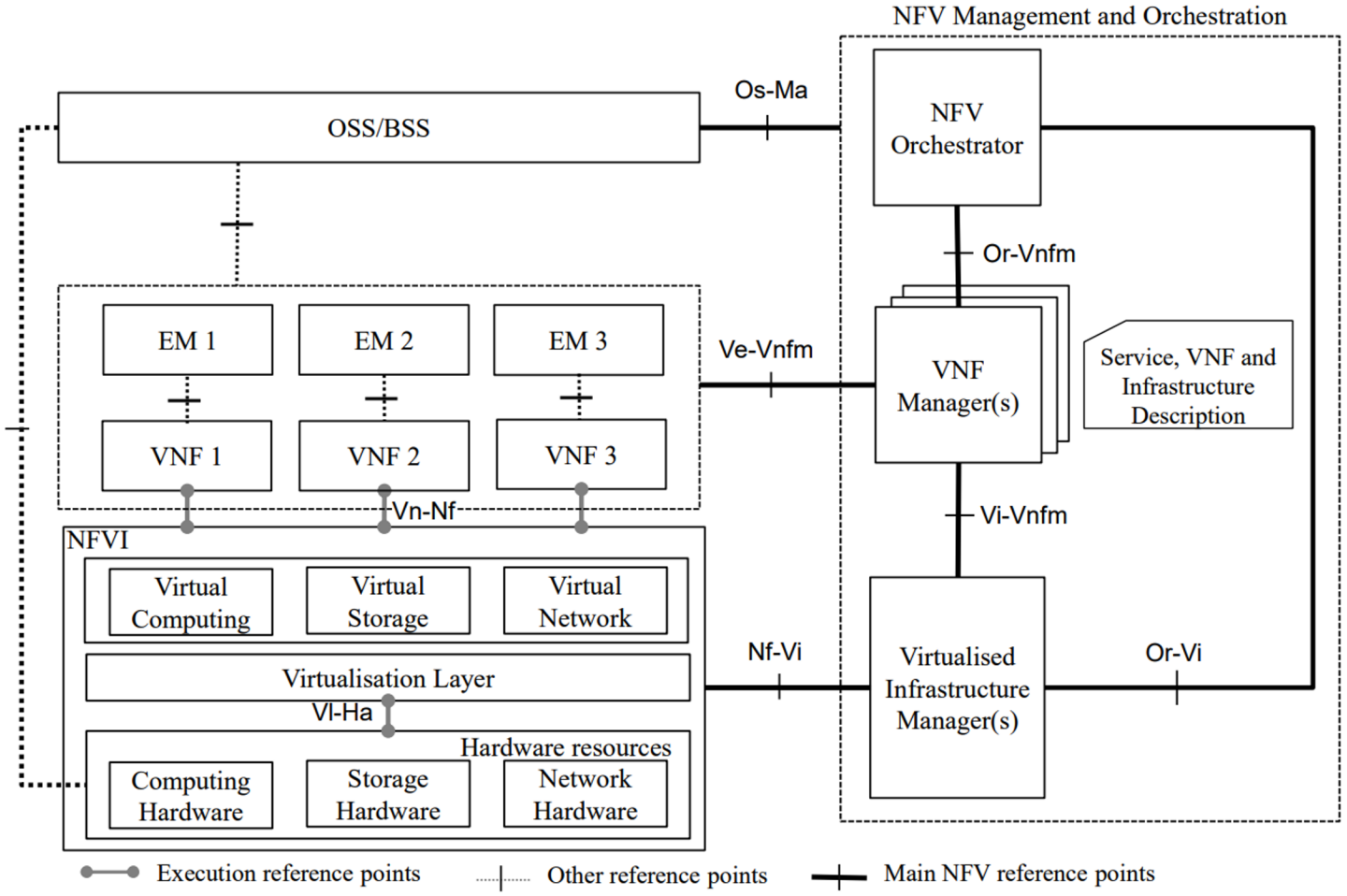
- NFV orchestration problems
  - Selection
    - Which VNFs are used and their ordering
  - Placement
    - Where in the physical network the VNFs will be instantiated
  - Chaining
    - Which virtual paths connect the placed VNFs



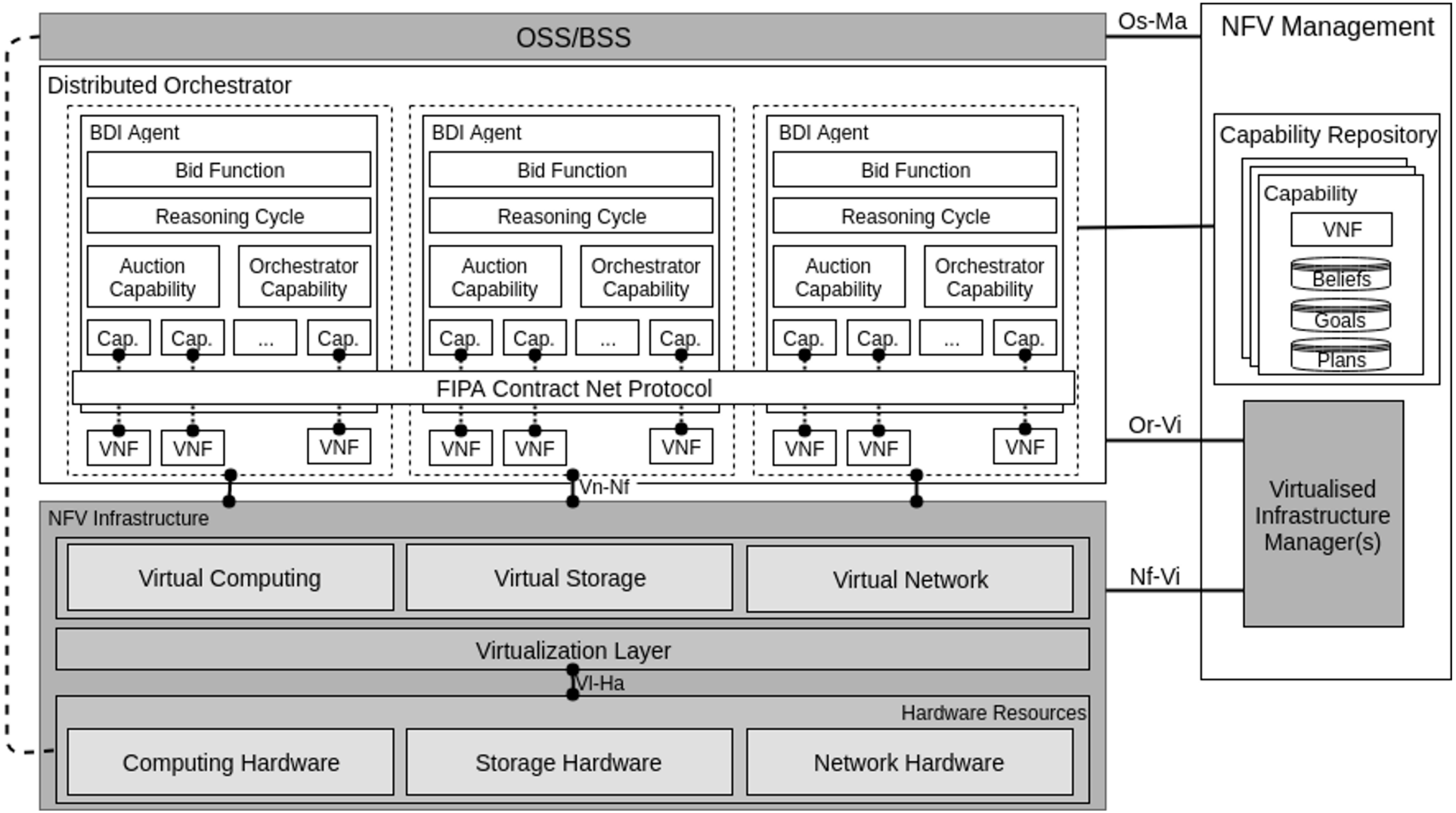
NFV Orchestration with BDI Agents

- Research Problem
  - **How to address the selection, placement, and chaining problems using a decentralised approach solution?**
- Proposed Solution
  - Decentralised NFV architecture that uses the BDI agents
  - Auction protocol to attack the selection, placement and chaining problems
  - Bidding heuristic
  - Implementation of the auction protocol and the bidding heuristic in a testbed

ETSI NFV Architecture

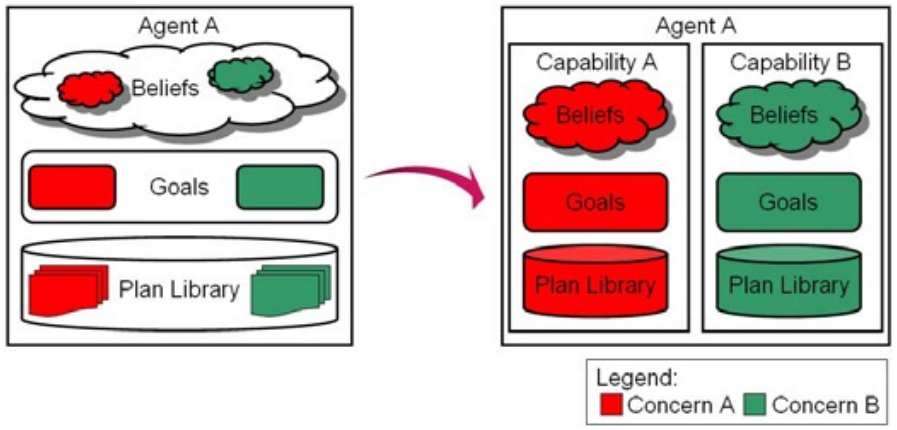


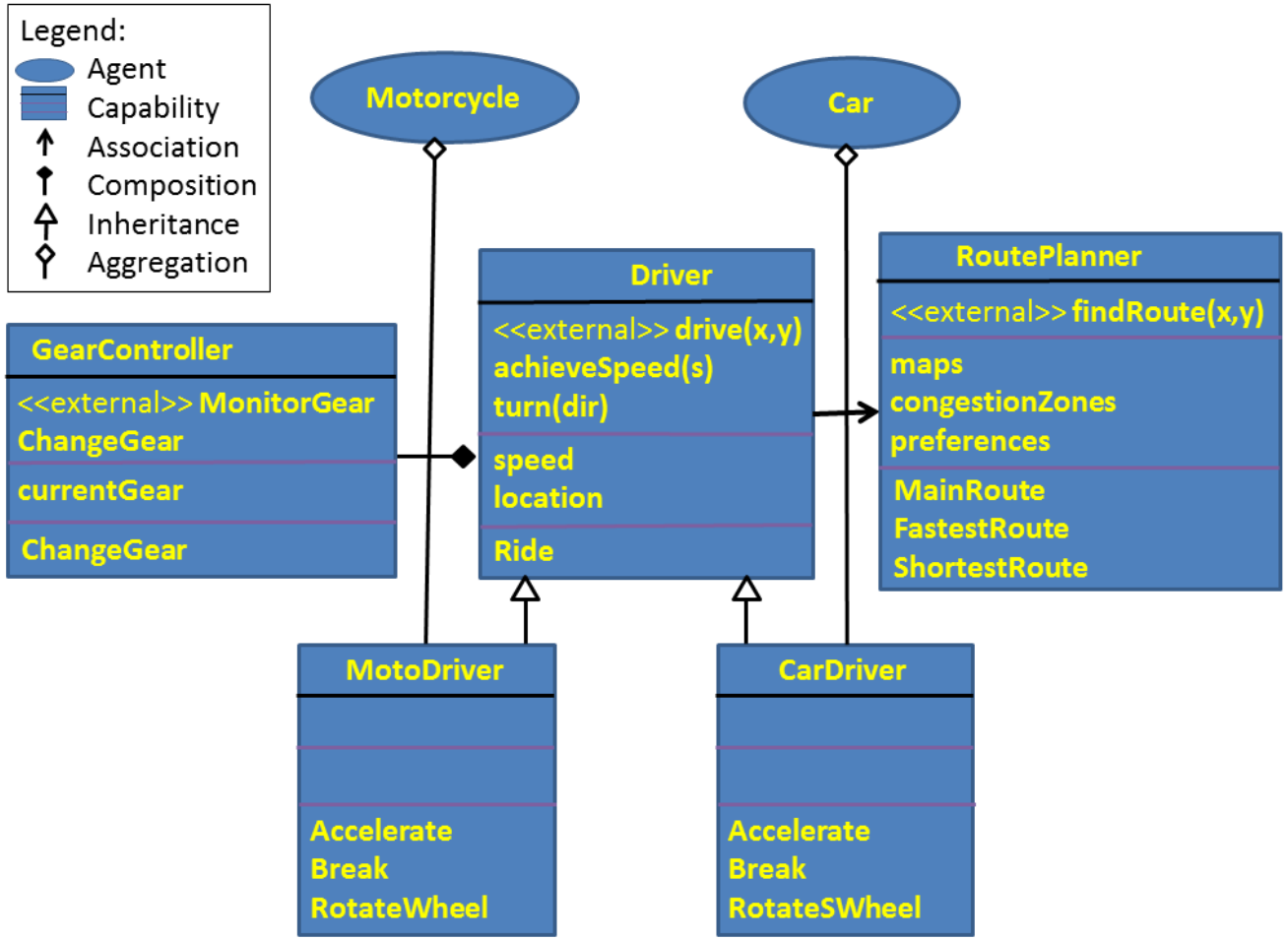
ETSI NFV Architecture



Capabilities and their Relationships

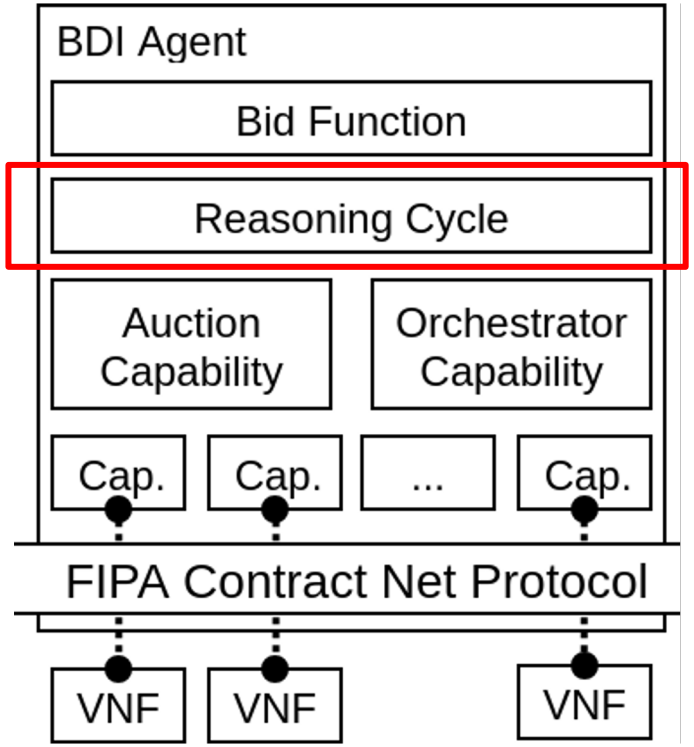
- Original Relationship
  - Inclusion
- Our Extensions
  - Association
  - Composition
  - Generalisation





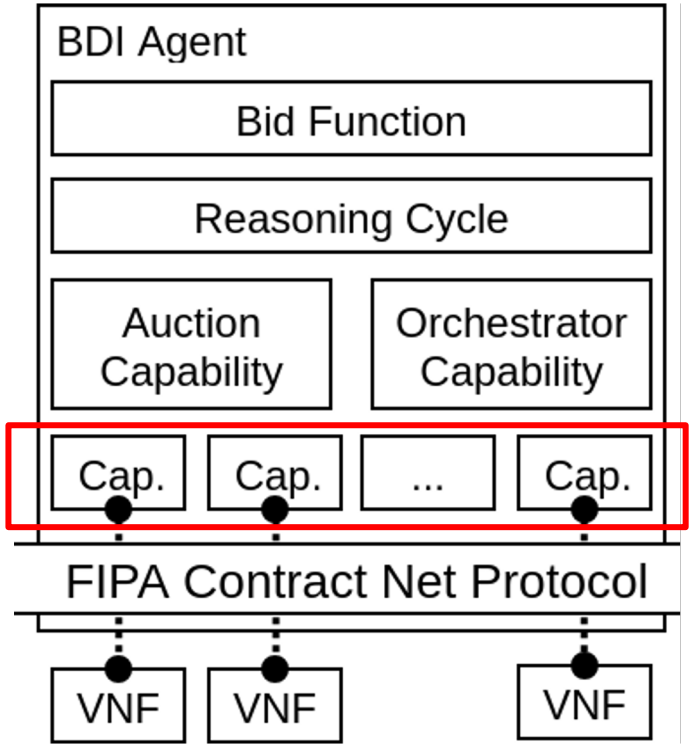
BDI Agents for NFV Orchestration

- Reasoning Cycle
  - The BDI reasoning cycle



BDI Agents for NFV Orchestration

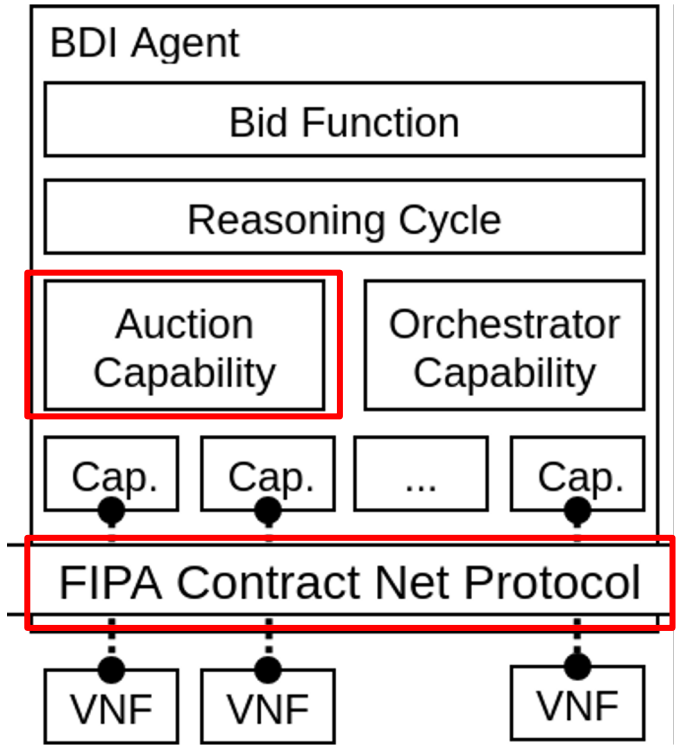
- Reasoning Cycle
  - The BDI reasoning cycle
- Capabilities
  - BDI agent modules
  - Cluster a set of beliefs and plans that together are able to handle events or achieve goals
  - Modularises a particular functional behaviour
    - Enables reuse



BDI Agents for NFV Orchestration

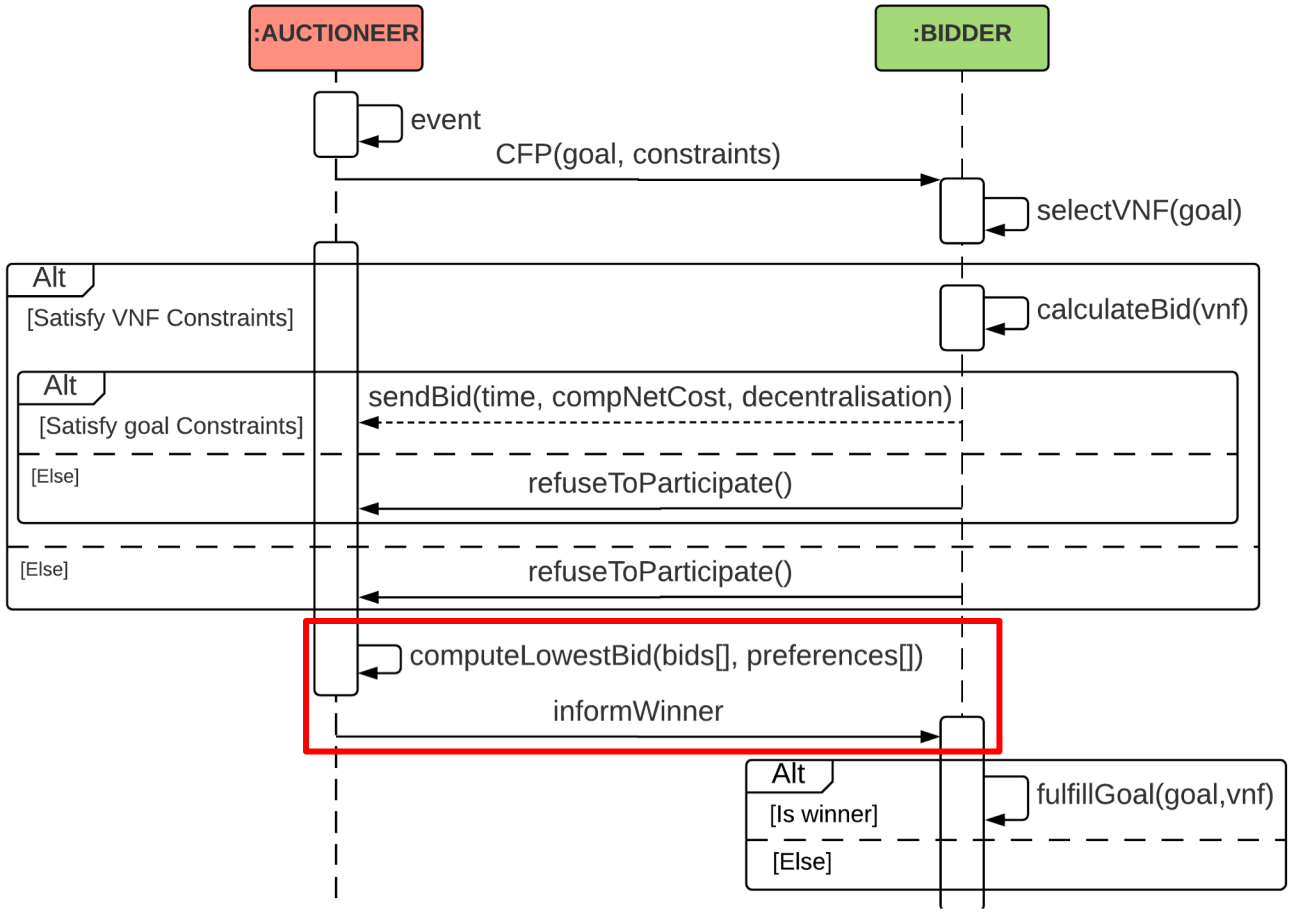
- Reasoning Cycle
  - The BDI reasoning cycle
- Auction Capability
  - Agents negotiate through **auctions** to attack the selection, placement and chaining problems

Auction algorithms are simple mechanisms used to reach agreements on the allocation of scarce resources



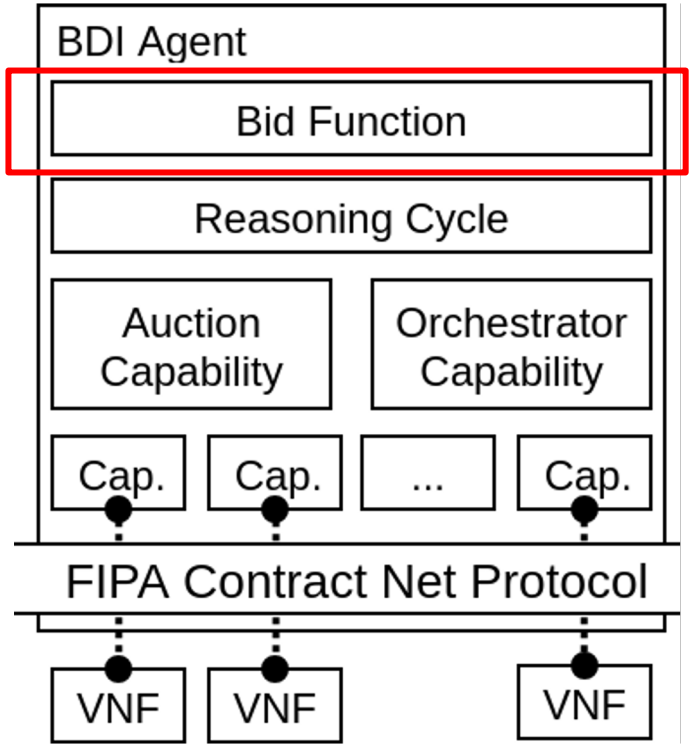
BDI Agents for NFV Orchestration

- Auction Process
  - NFV-A protocol
  - The bid with the **lowest cost**, considering the auctioneer's **preferences**, **wins** the auction



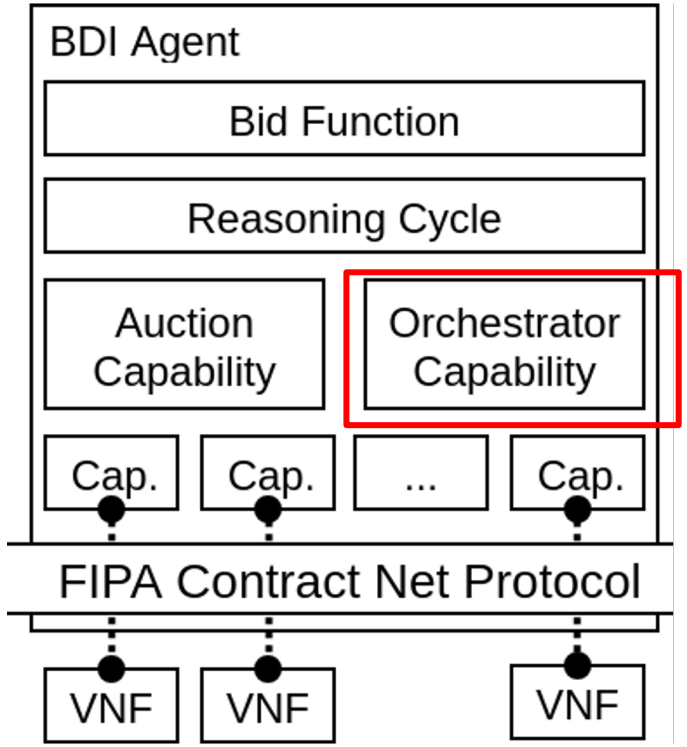
BDI Agents for NFV Orchestration

- Reasoning Cycle
  - The BDI reasoning cycle
- Auction Capability
  - Agents negotiate through auctions to attack the selection, placement and chaining problems
- Bid Function
  - Bidding heuristic



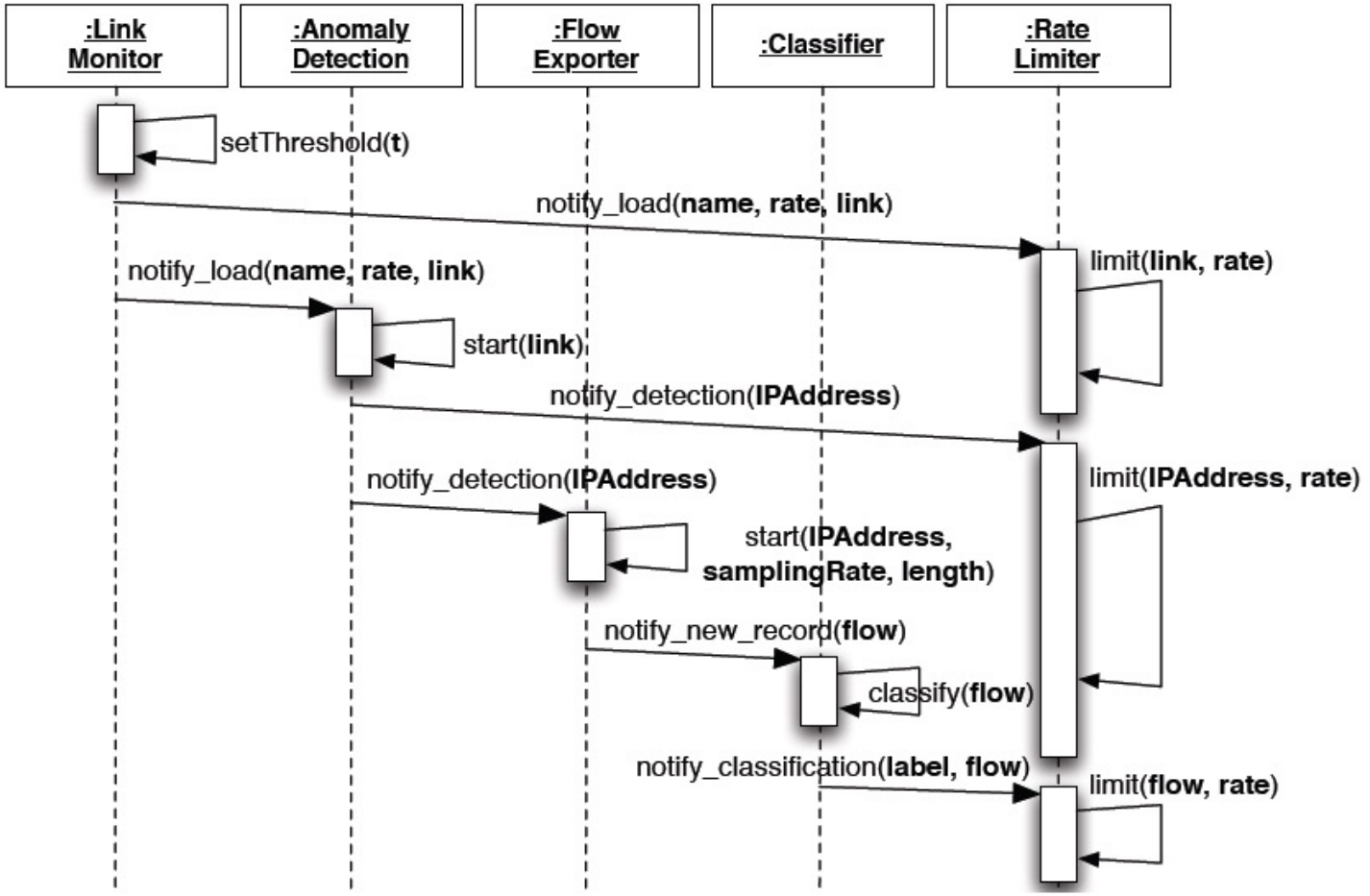
BDI Agents for NFV Orchestration

- Reasoning Cycle
  - The BDI reasoning cycle
- Auction Capability
  - Agents negotiate through auctions to attack the selection, placement and chaining problems
- Bid Function
  - Bidding heuristic
- Orchestrator Capability
  - Download VNFs, allocate resources and create virtual paths



Implementation and Evaluation

- BDI4JADE
  - Allow BDI agents to be implemented in pure Java
- Containernet
  - Fork of Mininet that provides OS-level virtualisation
- Agent-to-VNF communicatio
  - Each VNF is controlled by a Python script that receives/ makes RESTful calls from/to agents



BDI Agents to Combat a DDoS Attack

- Belief Revision
  - Detecting when the network is being **over used**
    - $usage(link) > threshold \rightarrow belief(overUsage(link))$
    - $usage(link) < threshold \rightarrow belief(overUsage(link))$
  - It is **unknown** that this **abnormal** behaviour is benign (not malicious)
    - $overUsage(link) \rightarrow belief(\sim regularUsage(link))$

BDI Agents to Combat a DDoS Attack

- Goal Generation and Deliberation
  - **Protect** the network and find out **if** there is an **attack**
    - $belief(overUsage(link)) \rightarrow goal(attackPrevented(link)) \wedge goal(?regularUsage(link))$
  - If there is a particular **IP** that is anomalous, achieve **similar** goals
    - $belief(anomalous(ip)) \rightarrow goal(restricted(ip)) \wedge goal(?benign(ip))$
  - If a **flow** is identified as a threat, **respond** this
    - $belief(threat(flow)) \rightarrow goal(threatResponded(flow))$

BDI Agents to Combat a DDoS Attack

- Goal Generation and Deliberation
  - Ideally, the network should **always** be **fully operational**
    - $belief\left(\neg fullyOperational(link)\right) \wedge belief\left(regularUsage(link)\right) \rightarrow goal\left(fullyOperational(link)\right)$
    - $belief\left(restricted(ip)\right) \wedge belief\left(\neg anomalous(ip)\right) \rightarrow goal\left(\neg restricted(ip)\right)$

BDI Agents to Combat a DDoS Attack

overUsage(link)	p
regularUsage(link)	~
fullyOperational(link)	
attackPrevented(link)	
anomalous(ip)	
restricted(ip)	
ipRateLimited(ip)	
benign(ip)	
flowRecord(ip)	
threat(flow)	
flowRateLimited(flow)	
threatResponded(flow)	

BDI Agents to Combat a DDoS Attack

		LLR
overUsage(link)	p	p
regularUsage(link)	~	~
fullyOperational(link)		¬
attackPrevented(link)		p
anomalous(ip)		
restricted(ip)		
ipRateLimited(ip)		
benign(ip)		
flowRecord(ip)		
threat(flow)		
flowRateLimited(flow)		
threatResponded(flow)		

Plan: LimitLinkRate (LLR)
Goal: $attackPrevented(link)$
Context: $overUsage(link)$
Actions:
$\texttt{limit}(link, rate)$
$belief(\neg fullyOperational(link))$
$belief(attackPrevented(link))$

BDI Agents to Combat a DDoS Attack

		LLR	ALS
overUsage(link)	p	p	¬*
regularUsage(link)	~	~	¬
fullyOperational(link)		¬	¬
attackPrevented(link)		p	p
anomalous(ip)			p
restricted(ip)			
ipRateLimited(ip)			
benign(ip)			
flowRecord(ip)			
threat(flow)			
flowRateLimited(flow)			
threatResponded(flow)			

Plan: AnalyseLinkStatistics (ALS)
Goal: $?regularUsage(link)$
Context: -
Actions:
// Analyse link and detect outliers
$\forall ip.(outlier(ip)) \longrightarrow belief(anomalous(ip)) \wedge belief(\sim benign(ip))$
$\exists ip.(anomalous(ip)) \longrightarrow belief(\neg regularUsage(link))$
$\nexists ip.(anomalous(ip)) \longrightarrow belief(regularUsage(link))$

BDI Agents to Combat a DDoS Attack

		LLR	ALS	LIPR
overUsage(link)	p	p	¬*	¬
regularUsage(link)	~	~	¬	p
fullyOperational(link)		¬	¬	¬
attackPrevented(link)		p	p	p
anomalous(ip)			p	p
restricted(ip)				p
ipRateLimited(ip)				
benign(ip)			~	
flowRecord(ip)				
threat(flow)				
flowRateLimited(flow)				
threatResponded(flow)				

Plan: LimitIPRate (LIPR)
Goal: $restricted(ip)$
Context: $anomalous(ip)$
Actions:
$\texttt{limit}(ip, rate)$
$belief(ipRateLimited(ip))$
$belief(restricted(ip))$
$\nexists ip'.(anomalous(ip') \wedge \neg restricted(ip')) \longrightarrow regularUsage(link)$

BDI Agents to Combat a DDoS Attack

		LLR	ALS	LIPR	RLR
overUsage(link)	p	p	¬*	¬	¬
regularUsage(link)	~	~	¬	p	p
fullyOperational(link)		¬	¬	¬	p
attackPrevented(link)		p	p	p	~
anomalous(ip)			p	p	p
restricted(ip)				p	p
ipRateLimited(ip)				p	p
benign(ip)			~	~	~
flowRecord(ip)					
threat(flow)					
flowRateLimited(flow)					
threatResponded(flow)					

Plan: RestoreLinkRate (RLR)
Goal: $fullyOperational(link)$
Context: $regularUsage(link)$
Actions:
$\texttt{unlimit}(link)$
$belief(fullyOperational(link))$
$belief(\sim attackPrevented(link))$

BDI Agents to Combat a DDoS Attack

		LLR	ALS	LIPR	RLR	AIPF(I)
overUsage(link)	p	p	¬*	¬	¬	¬
regularUsage(link)	~	~	¬	p	p	p
fullyOperational(link)		¬	¬	¬	p	p
attackPrevented(link)		p	p	p	~	~
anomalous(ip)			p	p	p	p
restricted(ip)						
ipRateLimited(ip)						
benign(ip)			~			
flowRecord(ip)						
threat(flow)						
flowRateLimited(flow)						
threatResponded(flow)						

Plan: AnalyseIPFlows (AIPF)
Goal: $?benign(ip)$
Context: $anomalous(ip)$
Actions:
$goal(?flowRecord(ip))$
// Classify flow record using machine learning
$\forall flow.(malicious(flow)) \longrightarrow belief(threat(flow))$
$\exists flow.(threat(flow) \wedge srcIP(flow) = ip) \longrightarrow belief(\neg benign(ip))$
$\nexists flow.(threat(flow) \wedge srcIP(flow) = ip) \longrightarrow belief(benign(ip))$

BDI Agents to Combat a DDoS Attack

		LLR	ALS	LIPR	RLR	AIPF(1)	RF
overUsage(link)	p	p	¬*	¬	¬	¬	¬
regularUsage(link)	~	~	¬	p	p	p	p
fullyOperational(link)		¬	¬	¬	p	p	p
attackPrevented(link)		p	p	p	~	~	~
anomalous(ip)			p	p	p	p	p
restricted(ip)				p	p	p	p
ipRateLimited(ip)				p	p	p	p
benign(ip)			~	~	~	~	~
flowRecord(ip)							p
threat(flow)							
flowRateLimited(flow)							
threatResponded(flow)							

Plan: RecordFlow (RF)
Goal: $flowRecord(ip)$
Context: -
Actions:
`recordFlow`(ip)
$belief(flowRecord(ip))$

BDI Agents to Combat a DDoS Attack

		LLR	ALS	LIPR	RLR	AIPF(1)	RF	AIPF(2)
overUsage(link)	p	p	¬*					
regularUsage(link)	~	~	¬					
fullyOperational(link)		¬	¬					
attackPrevented(link)		p	p					
anomalous(ip)			p					
restricted(ip)								
ipRateLimited(ip)				p	p	p	p	p
benign(ip)			~	~	~	~	~	¬
flowRecord(ip)							p	p
threat(flow)								p
flowRateLimited(flow)								
threatResponded(flow)								

Plan: AnalyseIPFlows (AIPF)
Goal: $?benign(ip)$
Context: $anomalous(ip)$
Actions:
$goal(?flowRecord(ip))$
// Classify flow record using machine learning
$\forall flow.(malicious(flow)) \longrightarrow belief(threat(flow))$
$\exists flow.(threat(flow) \wedge srcIP(flow) = ip) \longrightarrow belief(\neg benign(ip))$
$\nexists flow.(threat(flow) \wedge srcIP(flow) = ip) \longrightarrow belief(benign(ip))$

BDI Agents to Combat a DDoS Attack

		LLR	ALS	LIPR	RLR	AIPF(1)	RF	AIPF(2)	LFR
overUsage(link)	P	P	*						
regularUsage(link)	~	~							
fullyOperational(link)		¬							
attackPrevented(link)		P							
anomalous(ip)									
restricted(ip)				P	P	P	P	P	P
ipRateLimited(ip)				P	P	P	P	P	P
benign(ip)			~	~	~	~	~	¬	P
flowRecord(ip)							P	P	P
threat(flow)								P	~
flowRateLimited(flow)									P
threatResponded(flow)									P

Plan: LimitFlowRate (LFR)
Goal: $threatResponded(flow)$
Context: $threat(flow)$
Actions:
$\texttt{limit}(flow, rate)$
$belief(flowRateLimited(flow))$
$belief(threatResponded(flow))$
$belief(\sim threat(flow))$
$\nexists f.(threat(f) \wedge srcIP(flow) = srcIP(f)) \longrightarrow belief(benign(ip))$

BDI Agents to Combat a DDoS Attack

		LLR	ALS	LIPR	RLR	AIPF(1)	RF	AIPF(2)	LFR	RIPR
overUsage(link)	p	p	¬*	¬	¬	¬	¬	¬	¬	¬
regularUsage(link)	~	~	¬	p	p	p	p	p	p	p
fullyOperational(link)		¬	¬	¬	p	p	p	p	p	p
attackPrevented(link)		p	p	p	~	~	~	~	~	p
anomalous(ip)			p	p	p	p	p	p	p	~
restricted(ip)				p	p	p	p	p	p	¬
ipRateLimited(ip)							p	p	p	¬
benign(ip)							~	¬	p	p
flowRecord(ip)							p	p	p	p
threat(flow)								p	~	~
flowRateLimited(flow)									p	p
threatResponded(flow)									p	p

Plan: RestoreIPRate (RIPR)
Goal: $\neg restricted(ip)$
Context: $benign(ip) \wedge ipRateLimited(ip)$
Actions:
$\texttt{permit}(ip)$
$belief(\neg ipRateLimited(ip))$
$belief(\neg restricted(ip))$
$belief(\sim anomalous(ip))$

BDI Agents to Combat a DDoS Attack

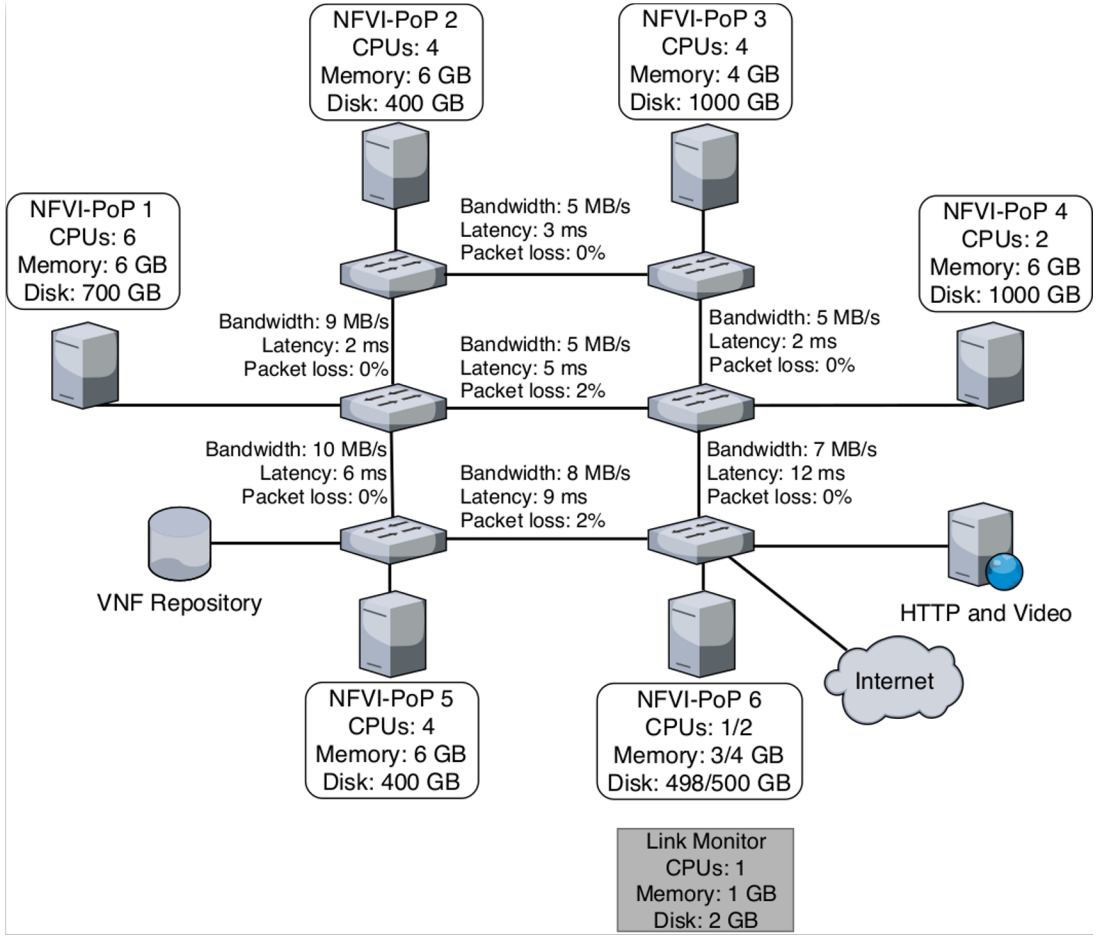
		LLR	ALS	LIPR	RLR	AIPF(1)	RF	AIPF(2)	LFR	RIPR
overUsage(link)	p	p	¬*	¬	¬	¬	¬	¬	¬	¬
regularUsage(link)	~	~	¬	p	p	p	p	p	p	p
fullyOperational(link)		¬	¬	¬	p	p	p	p	p	p
attackPrevented(link)		p	p	p	~	~	~	~	~	p
anomalous(ip)			p	p	p	p	p	p	p	~
restricted(ip)				p	p	p	p	p	p	¬
ipRateLimited(ip)				p	p	p	p	p	p	¬
benign(ip)			~	~	~	~	~	¬	p	p
flowRecord(ip)							p	p	p	p
threat(flow)								p	~	~
flowRateLimited(flow)									p	p
threatResponded(flow)									p	p

BDI Agents to Combat a DDoS Attack

- Simulation Settings

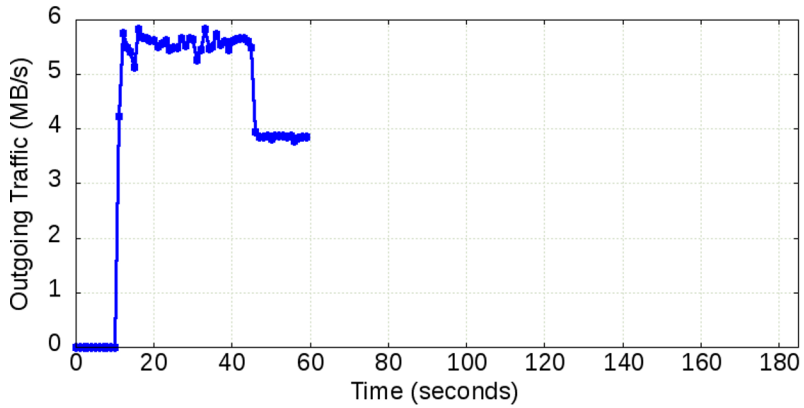
Parameter	Value
Number of servers	1 (HTTP and Streaming)
Number of non-malicious hosts	50
Traffic profile	Video: 17%, Web: 83%
Number of malicious hosts	8

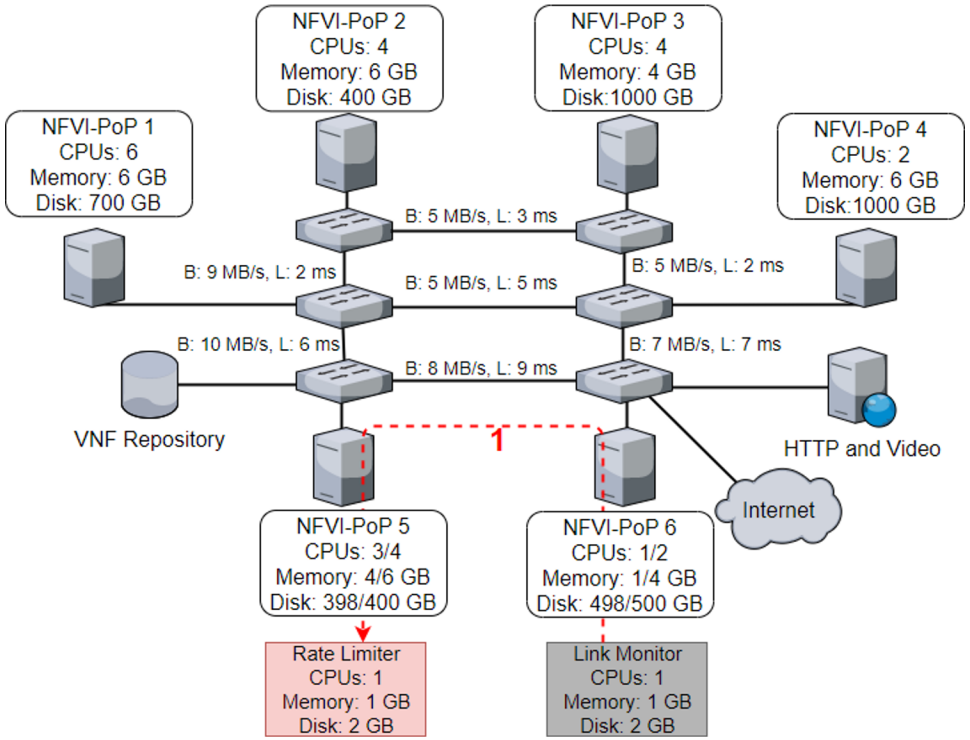
VNF	CPU	Memory	Disk	Bandwidth
Link Monitor	1 core	1 GB	2 GB	-
Rate Limiter	1 core	1 GB	2 GB	-
Anomaly Detector	1 core	2 GB	10 GB	2 MB/s
Classifier	2 core	4 GB	10 GB	1 MB/s
Load Balancer	1 core	1 GB	2 GB	-



BDI Agents to Combat a DDoS Attack

- 1: linkRateLimited(link)





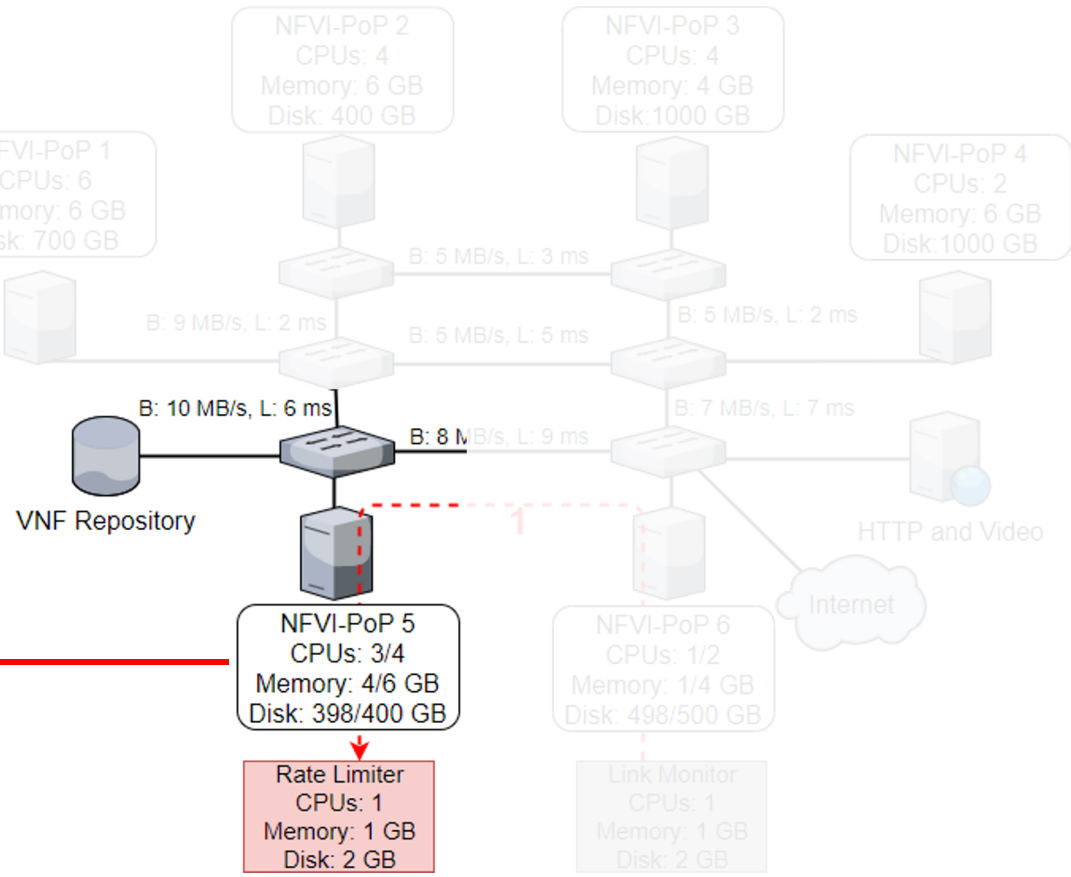
BDI Agents to Combat a DDoS Attack

Goal	$pf_{\mathcal{T}}$	$pf_{\mathcal{C}}$	$pf_{\mathcal{D}}$
linkRateLimited(link)	0.8	0.1	0.1

linkRateLimited(link)

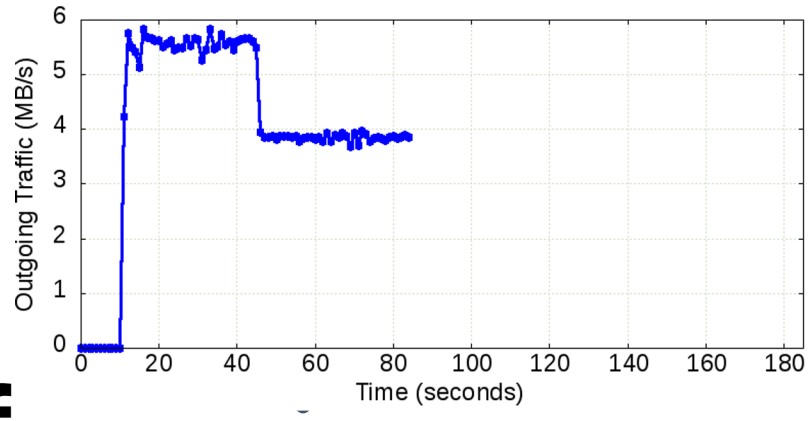
Agent	$\mathcal{T}$	$\mathcal{C}$	$\mathcal{D}$	Bid
1	50.00	16.88	25.84	0.50
2	55.56	26.91	39.63	0.54
3	100.00	16.87	33.12	0.88
4	100.00	4.88	18.01	0.87
5	0.00	5.65	13.22	0.09
6	62.50	0.00	0.00	0.60

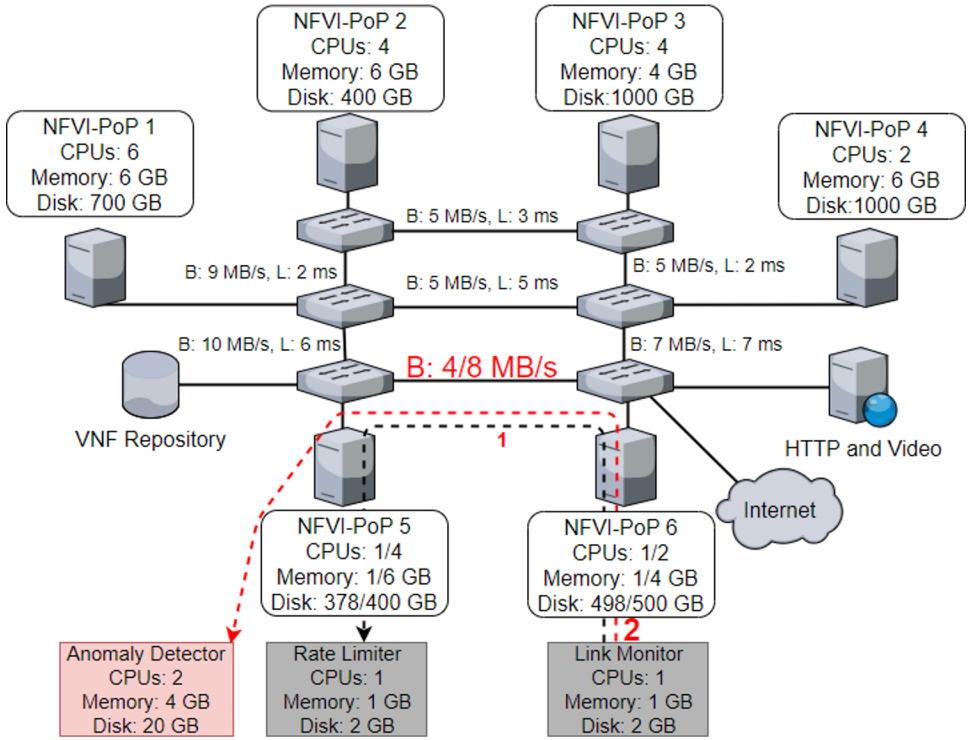
ipRateLimited(ip)



BDI Agents to Combat a DDoS Attack

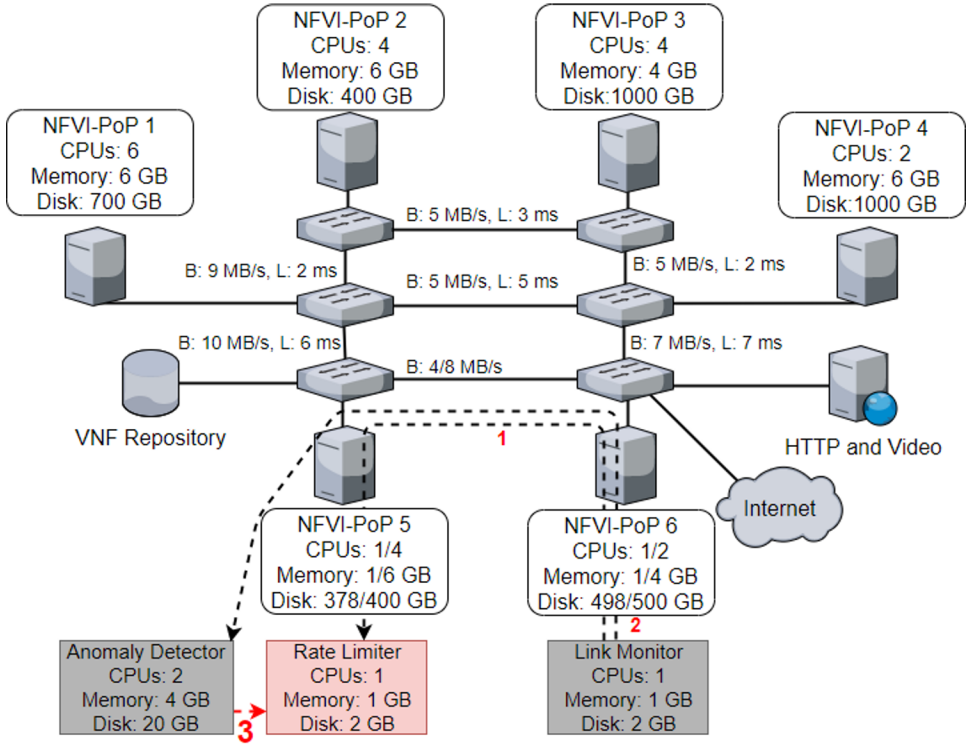
- 1: linkRateLimited(link)
- 2: ?anomalousUsage(link)

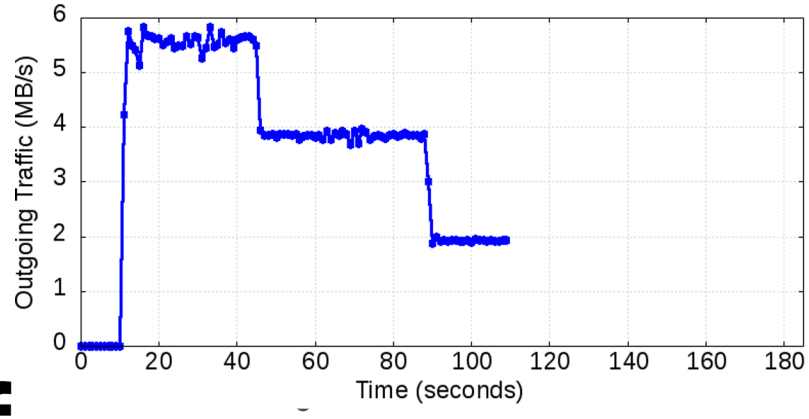




BDI Agents to Combat a DDoS Attack

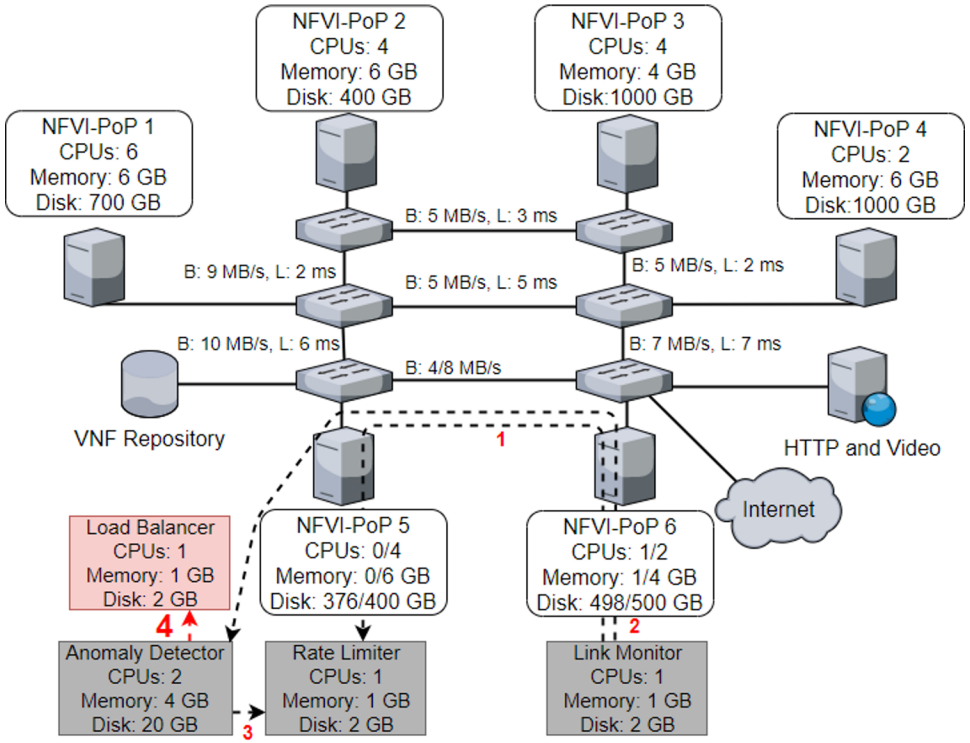
- 1: linkRateLimited(link)
- 2: ?anomalousUsage(link)
- 3: ipRateLimited(ip)

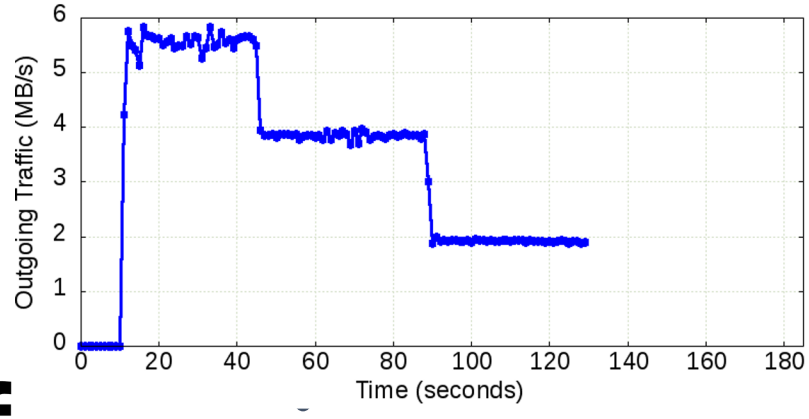




BDI Agents to Combat a DDoS Attack

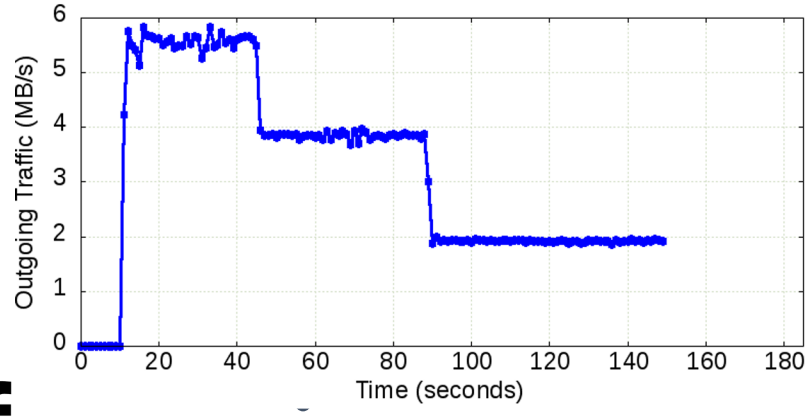
- 1: linkRateLimited(link)
- 2: ?anomalousUsage(link)
- 3: ipRateLimited(ip)
- 4: splitTrafficTowardsIp(ip)

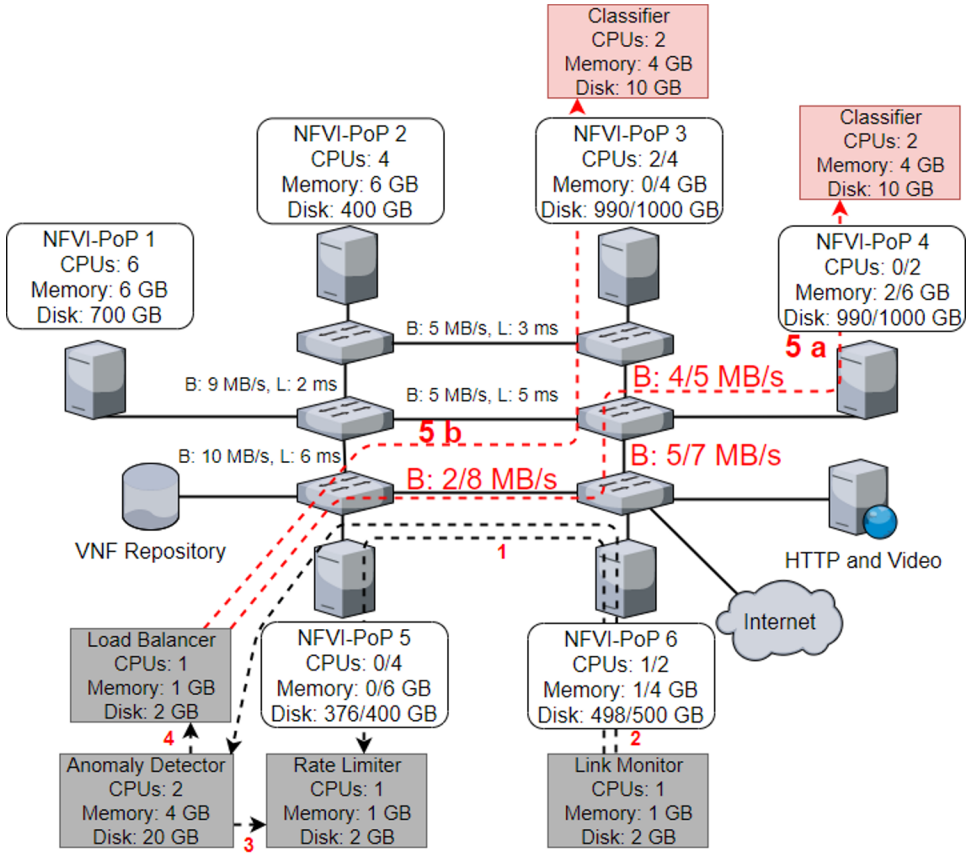




BDI Agents to Combat a DDoS Attack

- 1: linkRateLimited(link)
- 2: ?anomalousUsage(link)
- 3: ipRateLimited(ip)
- 4: splitTrafficTowardsIp(ip)
- 5a: flowRecord(ip)
- 5b: flowRecord(ip)



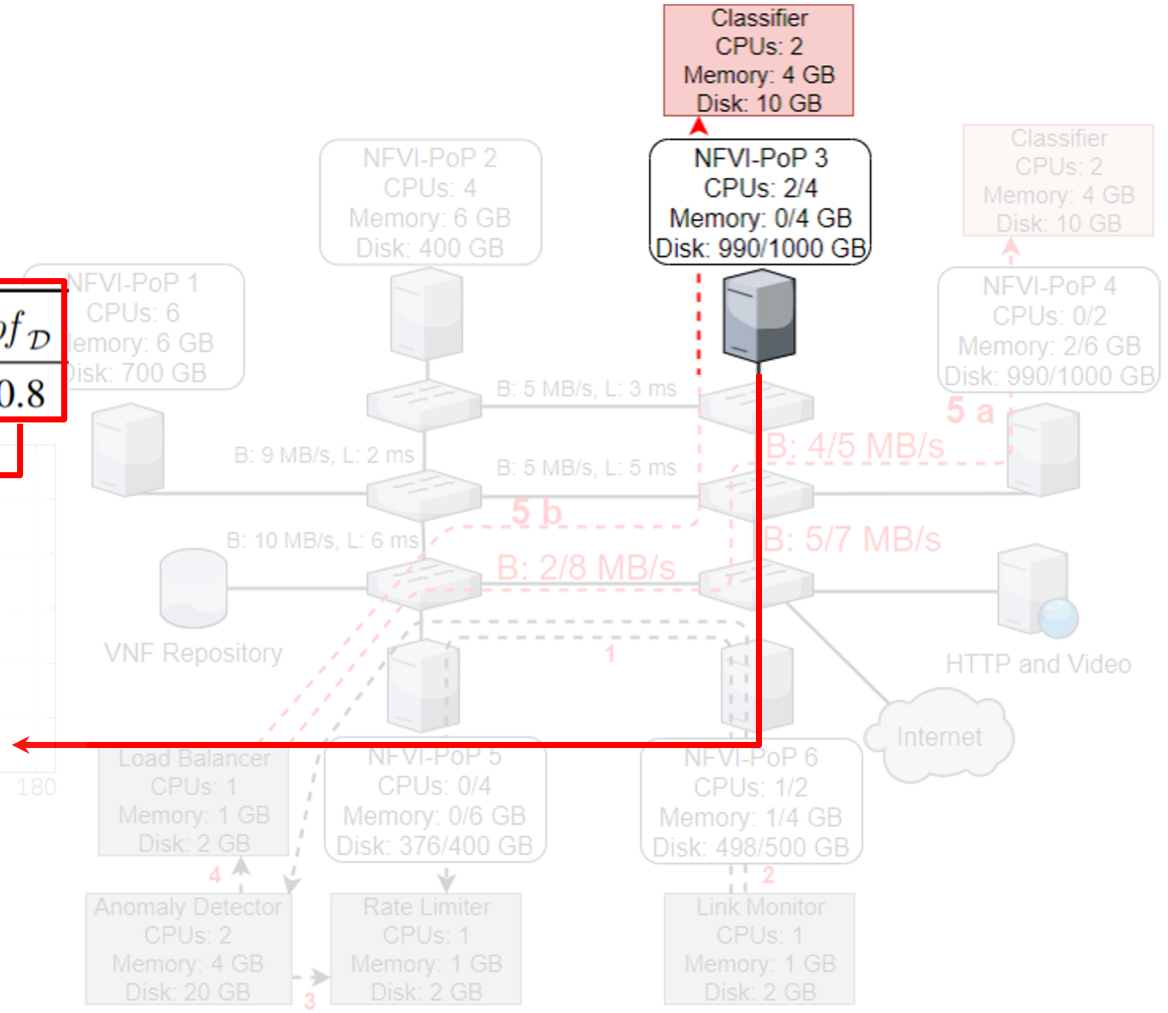


BDI Agents to Combat a DDoS Attack

Goal	$pf_{\mathcal{T}}$	$pf_{\mathcal{C}}$	$pf_{\mathcal{D}}$
flowRecord(ip)	0.0	0.2	0.8

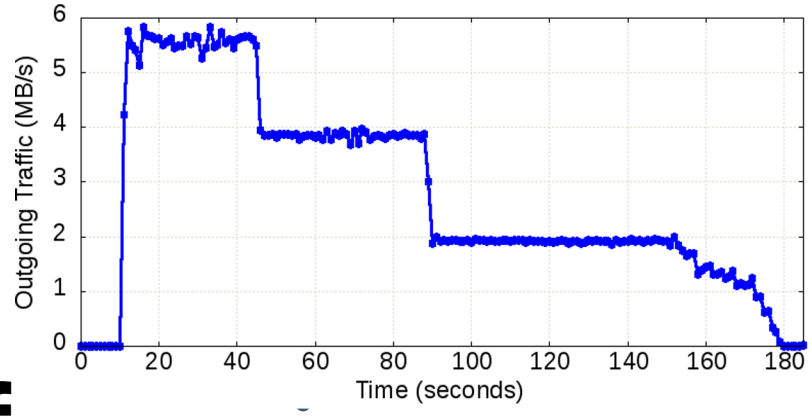
flowRecord(ip)

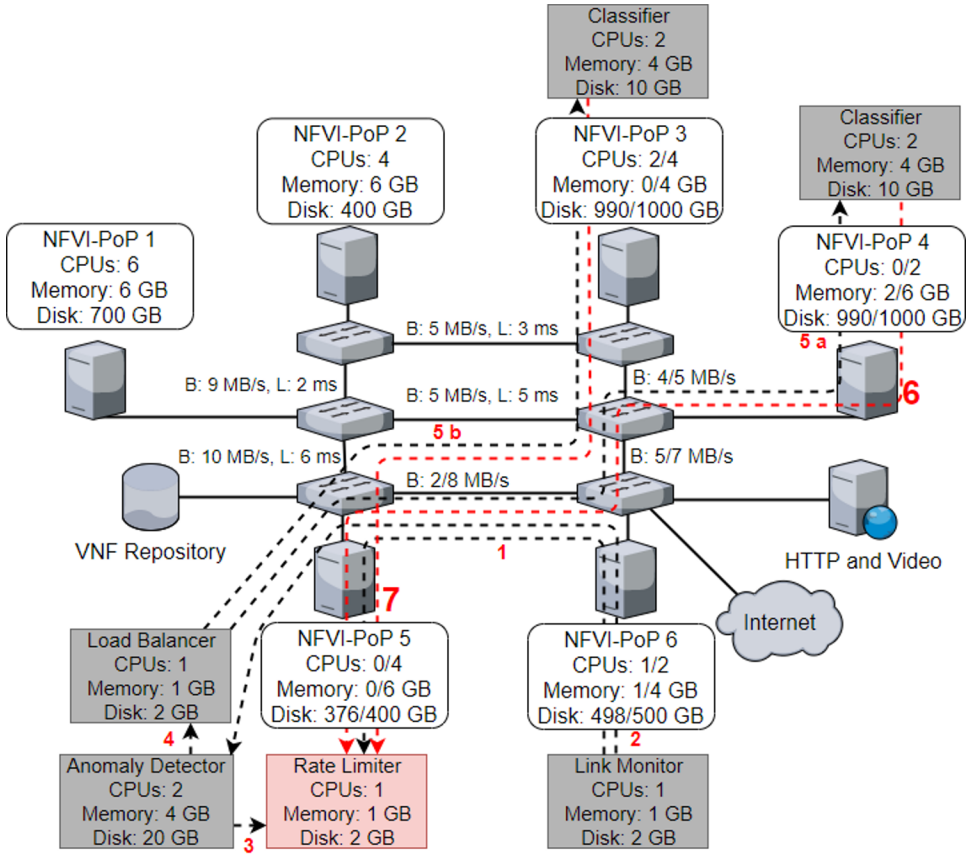
Agent	$\mathcal{T}$	$\mathcal{C}$	$\mathcal{D}$	Bid
1	50.00	0.40	11.04	0.80
2	55.56	19.62	13.06	0.92
3	100.00	24.61	52.69	0.20
4	100.00	14.61	39.15	0.38
5	-	-	-	-
6	-	-	-	-

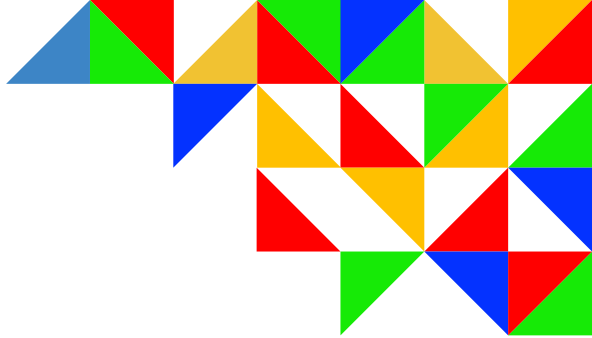


BDI Agents to Combat a DDoS Attack

- 1: linkRateLimited(link)
- 2: ?anomalousUsage(link)
- 3: ipRateLimited(ip)
- 4: splitTrafficTowardsIp(ip)
- 5a: flowRecord(ip)
- 5b: flowRecord(ip)
- 6 & 7: flowRateLimited(flow) x 8







Extensions to the BDI Architecture

Extensions of the BDI Architecture as Reusable Solutions to Ease the Development of Sophisticated BDI Agents

Extensions to the BDI Architecture

- **Problem**
  - How can a BDI agent **select** the best **plan** to achieve a goal based on its current **preferences** and the **uncertainty** of plan **outcomes**?
    - To be used by **mainstream** software **developers**

Model-driven Development

- Allows to **concentrate** in what is **important**
- May **remove** important **implementation details** in order to increase the productivity we need
  - Automation
  - Reuse
- Focus on business not code
- Key to building **complex software**

Extensions to the BDI Architecture

- **Problem**
  - How can a BDI agent **select** the best **plan** to achieve a goal based on its current **preferences** and the **uncertainty** of plan **outcomes**?
    - To be used by **mainstream** software **developers**
- **Our model-driven approaches**
  - Agents modelled based on an **extended BDI model**
  - Automatic **code generation** of agents able to select plans
- **Contributions**
  - **Meta-models** that extends the BDI model
  - **Algorithms** to **select plans**
  - **Transformation** of instances of our **meta-model** into source **code**

Plan Selection based on MAUT

Agent

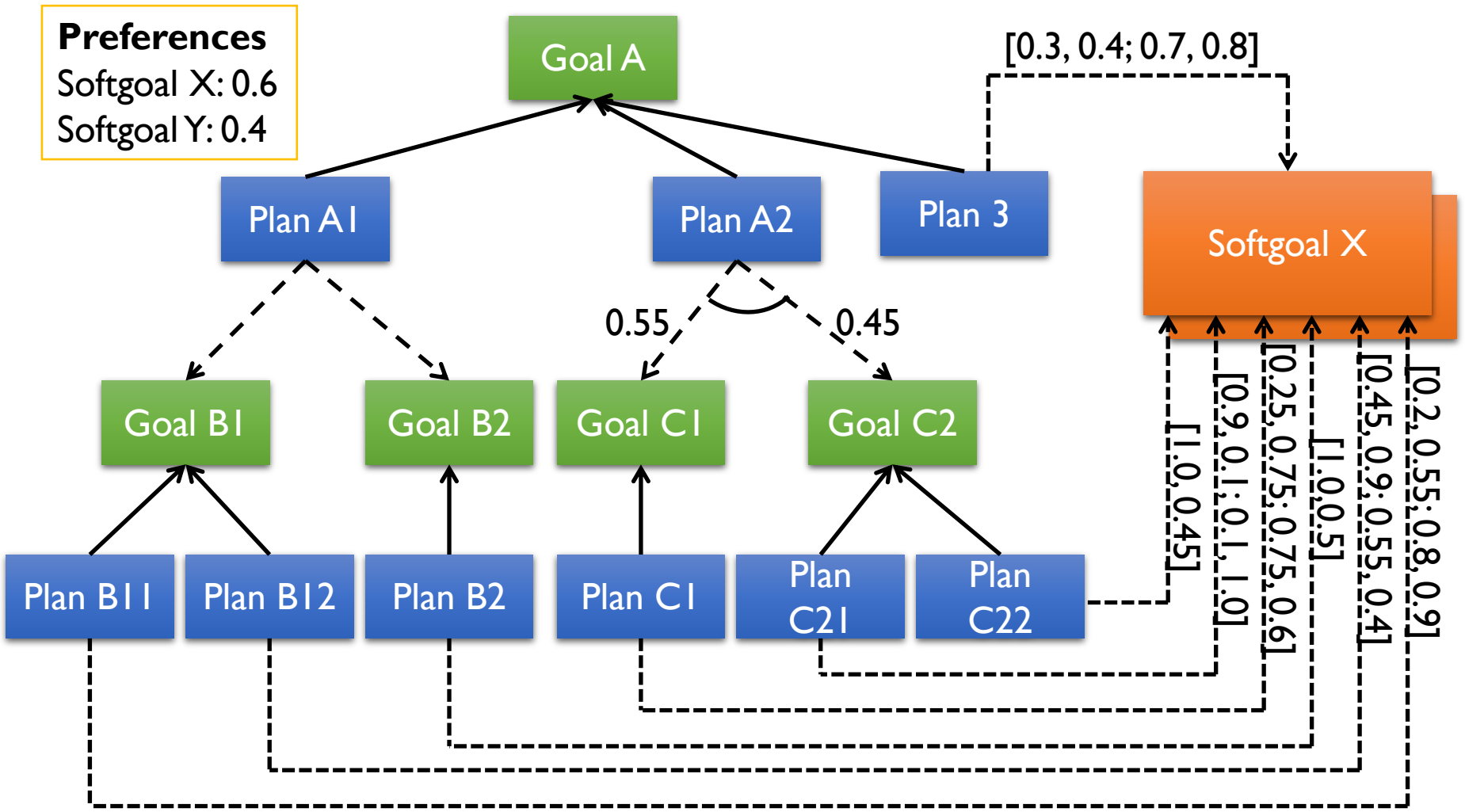
- << *B*, *G*, ***SG, P, Pref*** >>
  - B: beliefs
  - G: goals
  - **SG: softgoals**
  - **P: plans**
  - **Pref: preference function**

Softgoal (inspired by Tropos) is a broad agent objective, which is not achieved by a plan but is more or less satisfied due to the effect produced by agent actions that are part of plans.

Plan

- <<*G'*, **C, D**, *Body*>>
  - G': subset of goals
  - **C: contributions <<sg, prob, val>>**
    - **sg: softgoal**
    - **prob: probability**
      - **[0, 1]**
    - **val: contribution value**
      - **[0, 1]**
  - **D: dependencies**
    - **And Plan-Goal Dependency**
    - **Or Plan-Goal Dependency**
  - Body

Plan Selection based on MAUT



Plan Selection based on MAUT

- **Contributions** for softgoals may be **derived** from **dependencies**
- Each plan has
  - **Expected contribution** for each softgoal

$$EC(p, sg) = \sum_{c_i \in \mathcal{C} | c_i[sg] = sg} c_i[prob] \times c_i$$

  - **Plan Utility**

$$PU(p, Pref, SG) = \sum_{sg \in \mathcal{SG}} Pref(sg) \times$$

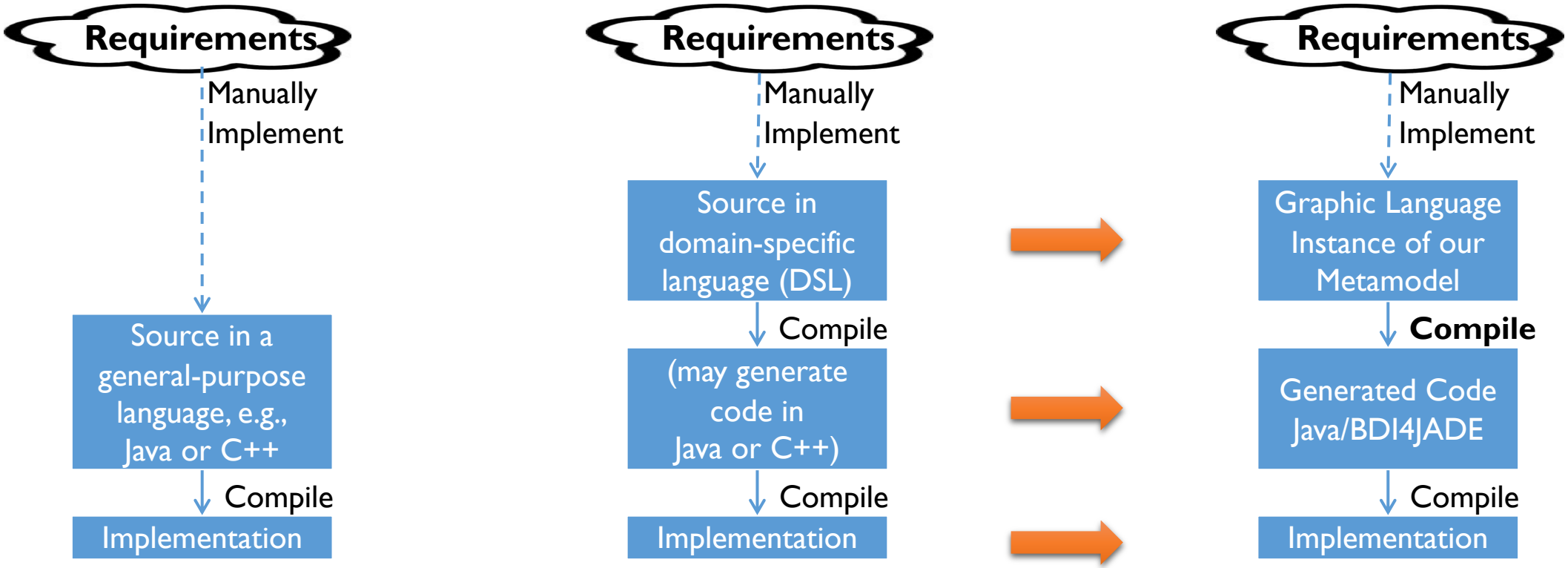
- Selected plan
  - **Maximum** utility

Algorithm 1: $SelectPlan(\mathcal{SG}, Pref, \mathcal{P})$

Input: $\mathcal{SG}$: set of agent softgoals, $Pref$: map of agent preferences; $\mathcal{P}$: set of plans that can achieve a goal
Output: $selectedP$: plan that best satisfies agent preferences

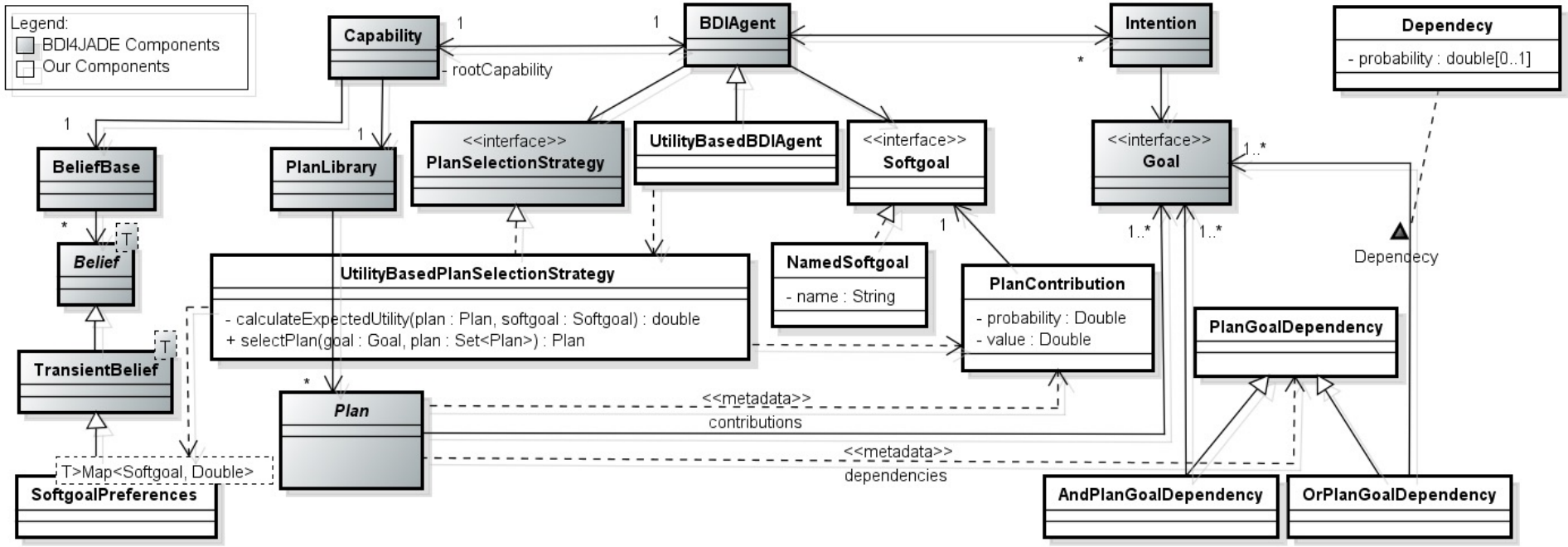
```
1  selectedP ← null;
2  maxContrib ← null;
3  foreach p ∈ P do
4      contrib ← 0;
5      foreach sg ∈ SG do
6          possibleContribs ← Contribution(p,sg);
7          expectedContrib ← 0;
8          foreach ⟨prob, value⟩ ∈ possibleContribs do
9              expectedContrib ← expectedContrib+prob×value;
10         contrib ← contrib + Pref(sg) × expectedContrib;
11         if selectedP = null ∨ maxContrib < contrib then
12             selectedP ← p;
13             maxContrib ← contrib;
14 return selectedPlan;
```

Model-to-Code Transformation



- Extension of **BDI4JADE**
- Use of the **Xpand2** template language
  - Part of the Model to Text (M2T) project of Eclipse

Model-to-Code Transformation



BDI 4JADE

Model-to-Code Transformation

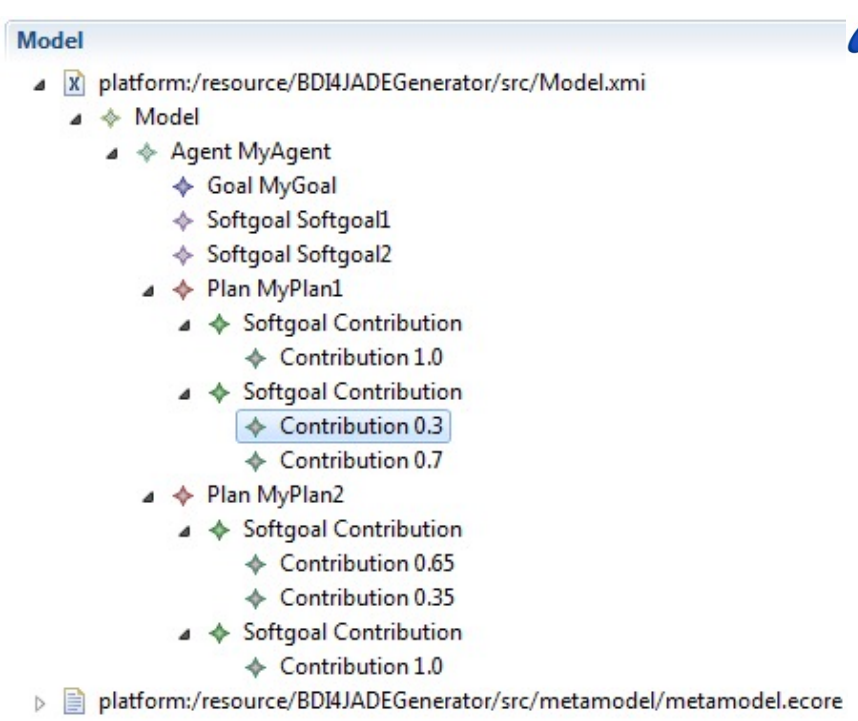
platform:/resource/BDI4JADEGenerator/src/metamodel/metamodel.ecore
- metamodel
  - Model
    - agents : Agent
  - Agent
    - name : EString
    - beliefBase : Belief
    - goals : Goal
    - softgoals : Softgoal
    - plans : Plan
  - Belief
  - Goal
  - Softgoal
    - name : EString
    - preference : EDouble
  - Plan
    - name : EString
    - goal : Goal
    - contributions : SoftgoalContribution
    - dependencies : PlanGoalDependency
  - SoftgoalContribution
    - softgoal : Softgoal
    - contributions : Contribution
  - Contribution
    - probability : EDouble
    - value : EDouble
  - PlanGoalDependency
  - AndPlanGoalDependency -> PlanGoalDependency
  - OrPlanGoalDependency -> PlanGoalDependency
  - Dependency

Metamodel

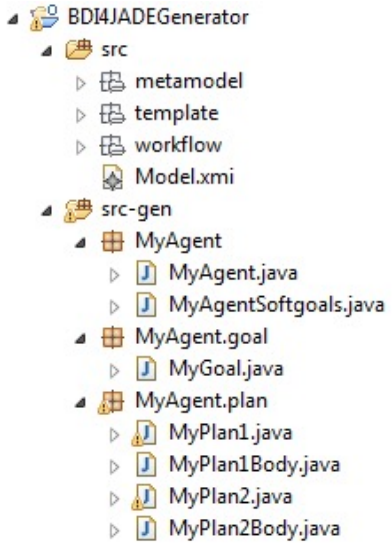
Template

```
«DEFINE planClass FOR Plan»
«FILE ((Agent)this.eContainer).name + "/plan/" + name + ".java"»
package «((Agent)this.eContainer).name».plan;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;

import bdi4jade.plan.PlanContribution;
import bdi4jade.softgoal.Softgoal;
import bdi4jade.util.plan.SimplePlan;

«EXPAND importSoftgoals FOR (Agent)this.eContainer»
«EXPAND importGoal FOR this.goal»

public class «this.name» extends SimplePlan {

    public «this.name»() {
        super(«this.goal.name».class, «this.name»Body.class);

        Map<Softgoal, List<PlanContribution>> contributions =
            (Map<Softgoal, List<PlanContribution>>) getMetadata(I [...]
        List<PlanContribution> sgContributions = null;

        // Setup contributions
        «EXPAND softgoalContribution FOREACH this.contributions»
    }

}
«ENDFILE»
«ENDDEFINE»

«DEFINE softgoalContribution FOR SoftgoalContribution»
sgContributions = new ArrayList<PlanContribution>();
«EXPAND contribution FOREACH this.contributions»
contributions.put(«((Agent)this.eContainer.eContainer).name»Softg [...]
«ENDDEFINE»

«DEFINE contribution FOR Contribution»sgContributions.add(new Pla [...]
```

Model-to-Code Transformation



```
package MyAgent.plan;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;

import bdi4jade.plan.PlanContribution;
import bdi4jade.softgoal.Softgoal;
import bdi4jade.util.plan.SimplePlan;

import MyAgent.MyAgentSoftgoals;
import MyAgent.goal.MyGoal;

public class MyPlan1 extends SimplePlan {

    public MyPlan1() {
        super(MyGoal.class, MyPlan1Body.class);

        Map<Softgoal, List<PlanContribution>> contributions =
            (Map<Softgoal, List<PlanContribution>>) getMetadata(DefaultMetadata.CONTRIBUTIONS);
        List<PlanContribution> sgContributions = null;

        // Setup contributions

        sgContributions = new ArrayList<PlanContribution>();
        sgContributions.add(new PlanContribution(MyAgentSoftgoals.Softgoal1,
                1.0, 0.6));
        contributions.put(MyAgentSoftgoals.Softgoal1, sgContributions);

        sgContributions = new ArrayList<PlanContribution>();
        sgContributions.add(new PlanContribution(MyAgentSoftgoals.Softgoal2,
                0.3, 1.0));
        sgContributions.add(new PlanContribution(MyAgentSoftgoals.Softgoal2,
                0.7, 0.45));
        contributions.put(MyAgentSoftgoals.Softgoal2, sgContributions);

    }

}
```



Evaluation

- **Scenario**
  - **Transportation**
    - Bike
    - Bus
    - Car
    - Motorcycle
  - **Softgoals**
    - Maximise Safety
    - Maximise Security
    - Maximise Performance
    - Minimise Cost
    - Maximise Comfort
  - **Probabilities** associated with
    - Time taken
    - Crashing
    - Being robbed

- **Experiment**
  - Randomly generate **preferences**
  - Randomly generate a **scenario**
  - Compute **satisfaction** for each transportation type
  - **Select** a **plan** using a plan selector
  - **Store** the satisfaction according to the selected plan

Evaluation

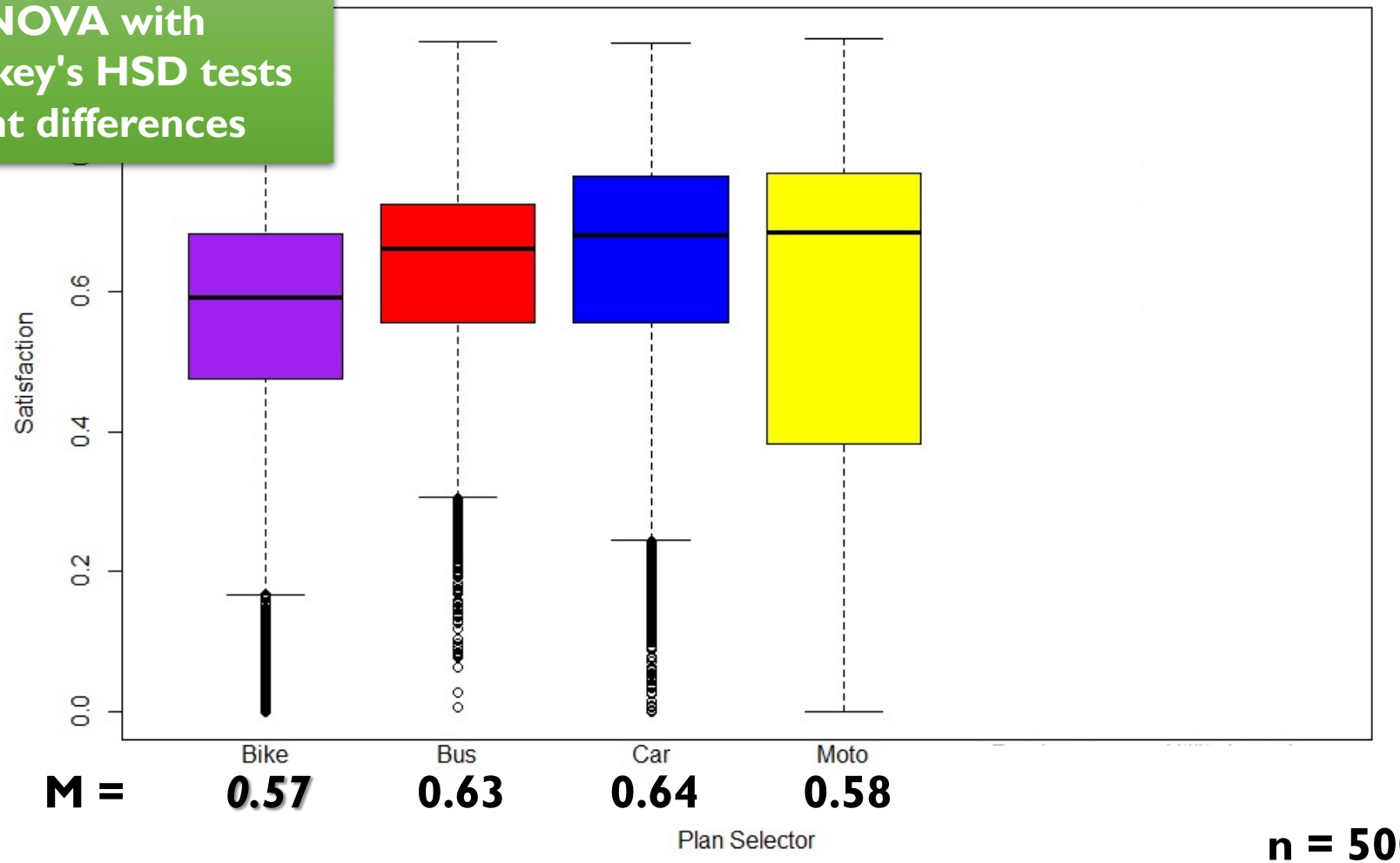
Softgoal	Bicycle	
	Prob	Val
Safety	*0.05*	0.00
	0.95	1.00
Security	*0.20*	0.00
	0.80	1.00
Performance	*0.05*	0.00
	0.475	0.00
	0.475	0.04
Cost	*0.05*	0.00
	0.95	0.90
Comfort	*1.00*	0.20

Probability of Being Robbed

Probability of Crashing

Evaluation: Results

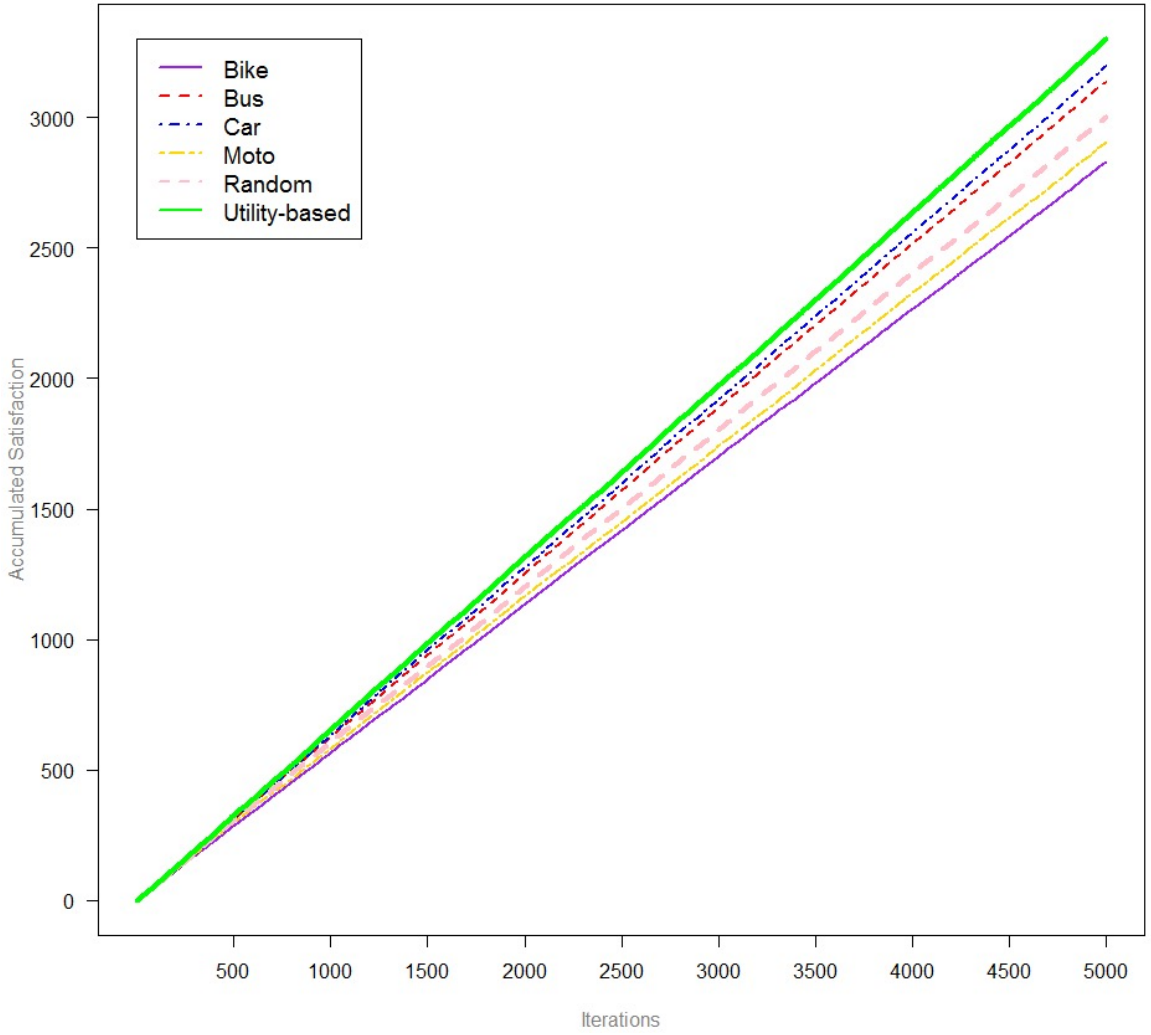
One-way ANOVA with Post-hoc Tukey's HSD tests → Significant differences



	Bike	Bus	Car	Moto
M =	***0.57***	**0.63**	**0.64**	**0.58**

n = 5000

Evaluation: Results



- **Best results**
- **Limitations**
  - Representation of dependent probabilities
  - Plans have the same set of contributions
  - Quantitative values for preferences

Learning-based Plan Selection

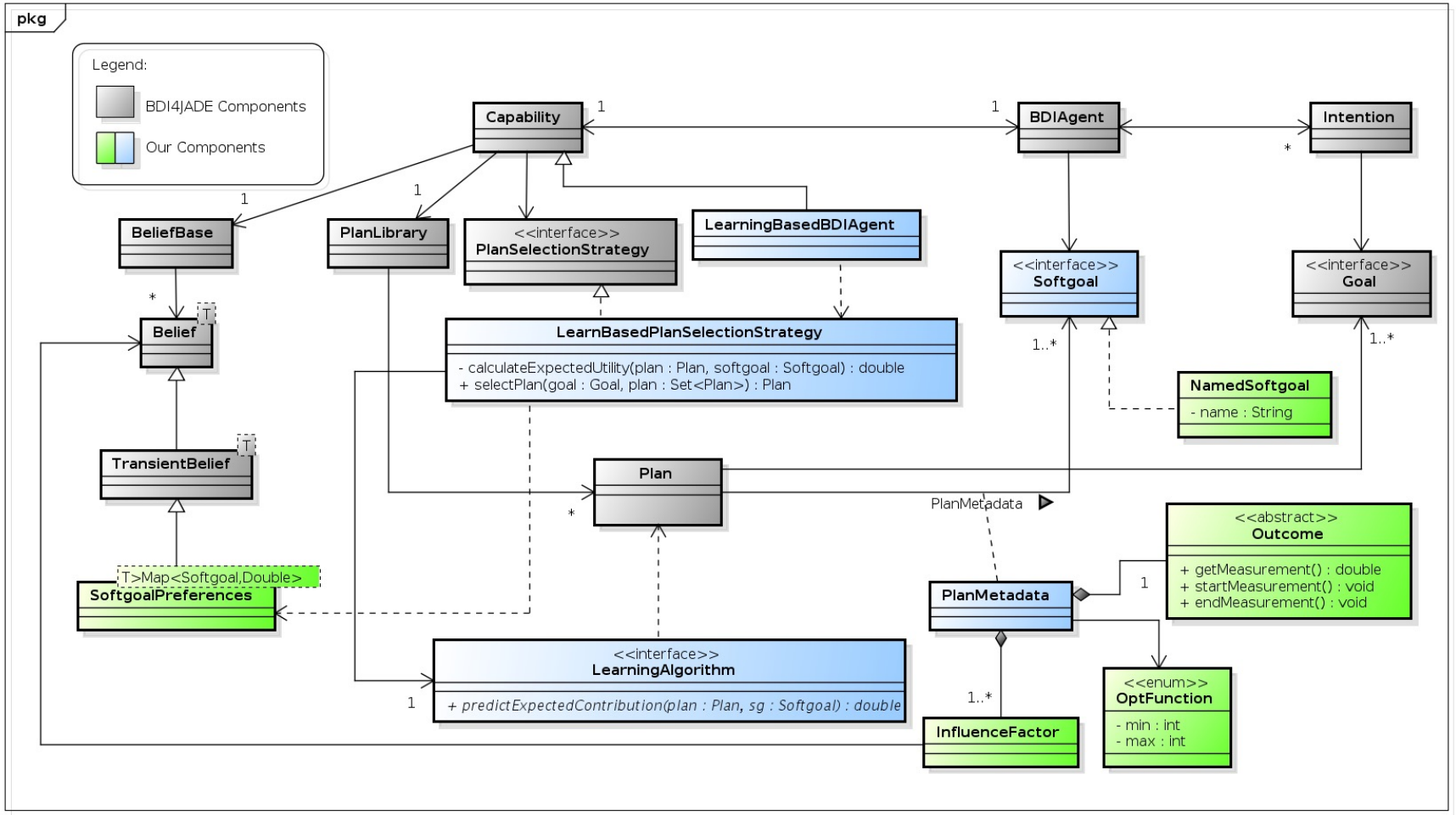
- Several plan-selection approaches have been proposed in this context
  - **Learning** the **context** in which a plan possibly fails
  - Identifying the plan that best matches agent's **preferences**
- They do not consider
  - **Uncertainty** involving plan executions
  - Rely on information that is **hard** to **elicit** at development time

How to **select** a **plan** considering the **context** in which an agent is inserted, taking into account agent's **preferences** and the **uncertainty** of plan executions?

Learning-based Plan Selection

- Agents **learn** which are the **expected** plan **outcomes** according to **particular contexts**
  - It makes them able to **predict plan outcomes** based on the current context
- Information used to **select** a **plan** to achieve a goal
  - Taking into account **preferences** that are expressed over agent's **softgoals**
- Approach components
  - A **meta-model**
  - A **technique** to learn and predict plan outcomes

Learning-based Plan Selection



* Implemented as an extension of the BDI4JADE platform

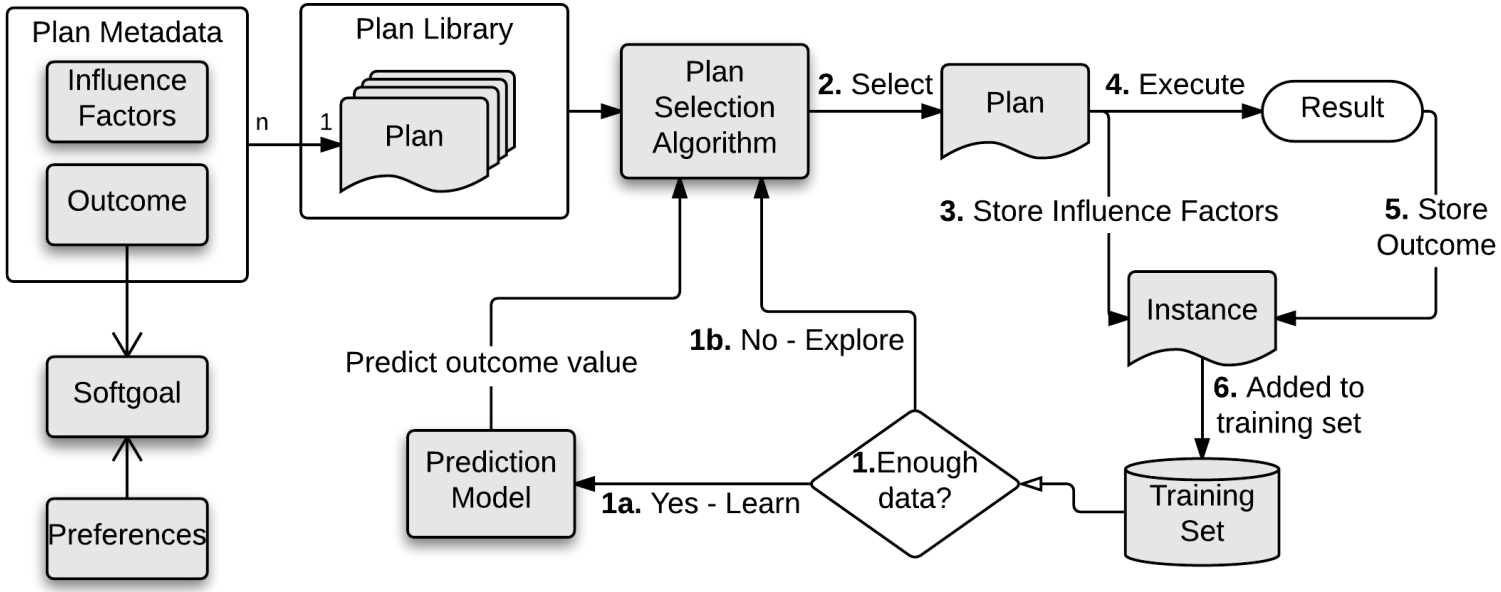
Learning-based Plan Selection

- Describes how to **use the information** provided by our **meta-model** in order to properly **select** a **plan** in a given context
- In an **initial** learning **stage**, plans are selected **randomly**
  - Information **collected** and **stored** from plan executions
  - **Threshold**

Execution	Truck Conditions	Traffic Conditions	Road Conditions	Time Taken
1	0.591	0.903	0.096	2.35
2	0.858	0.987	0.419	2.5
3	0.495	0.430	0.677	1.725
...	...	...	...	...

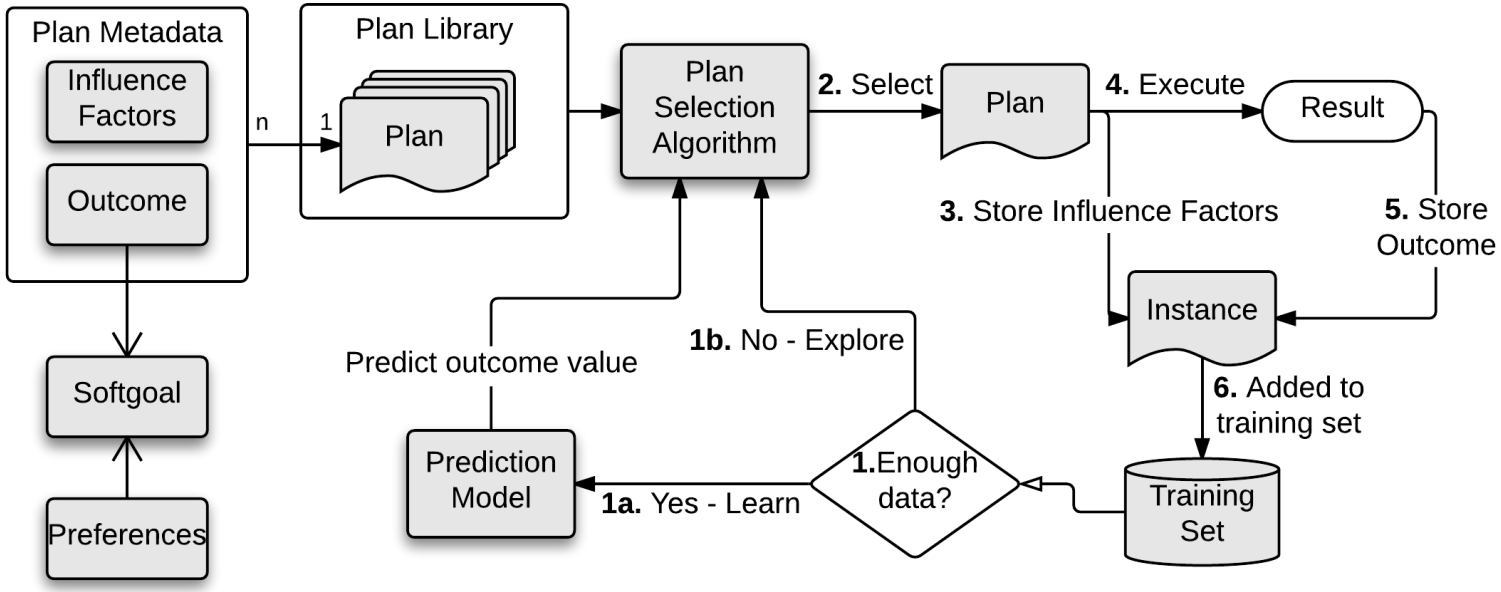
Learning-based Plan Selection

- Use of existing **machine learning algorithms** to predict outcomes values
  - **Linear regression**



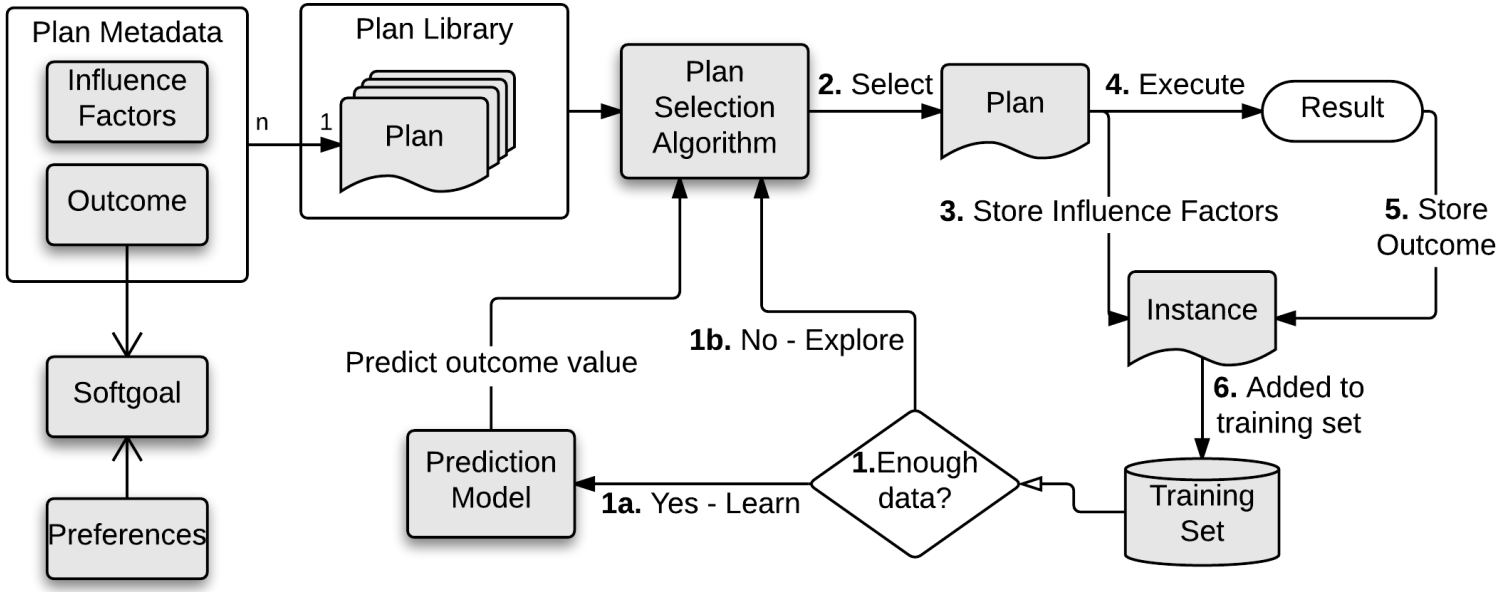
Learning-based Plan Selection

- Predicted outcomes are **converted** to an agent **utility**
  - Utilities are **combined**

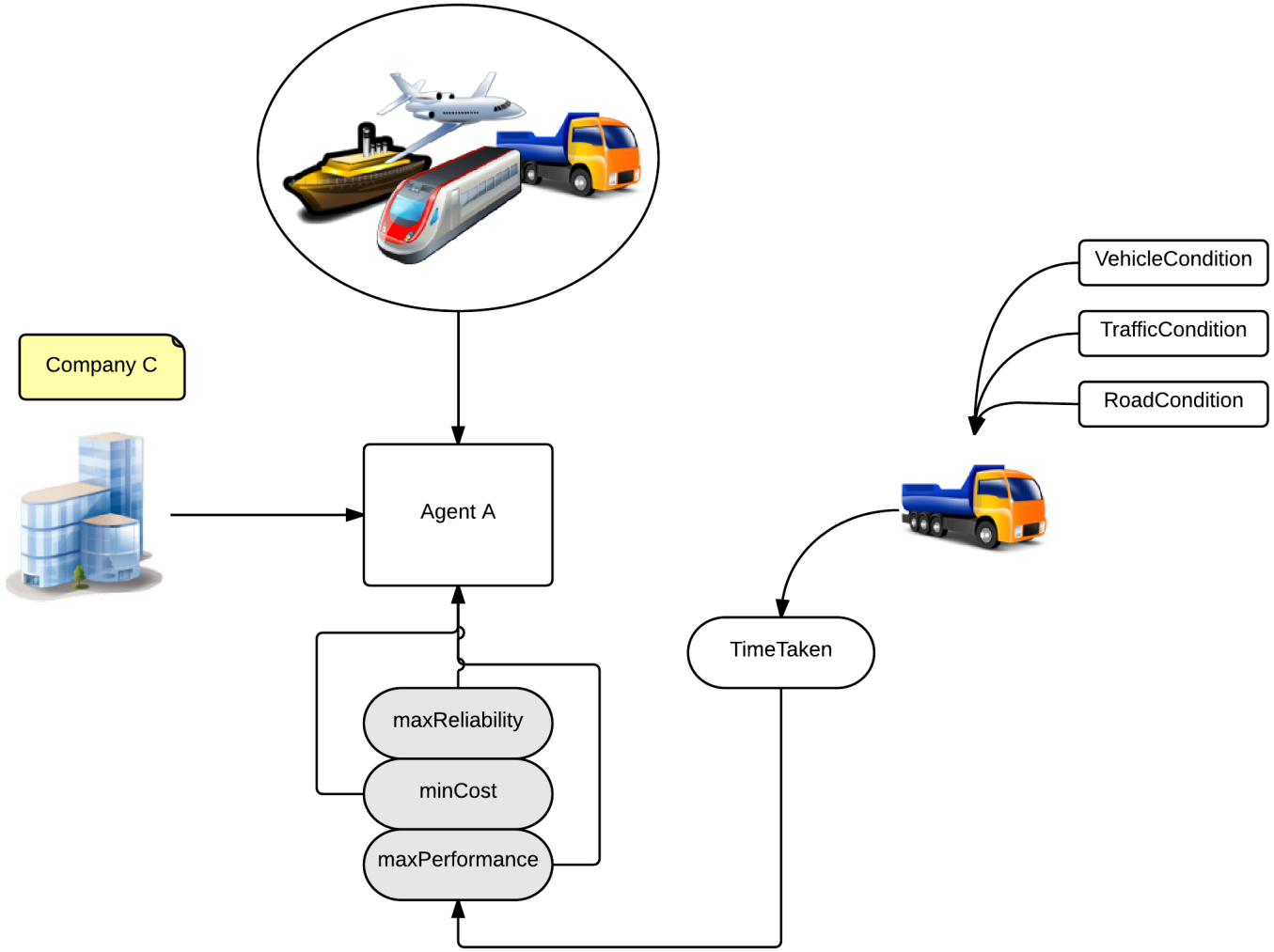


Learning-based Plan Selection

- Plan with the **highest predicted utility** is selected



Learning-based Plan Selection



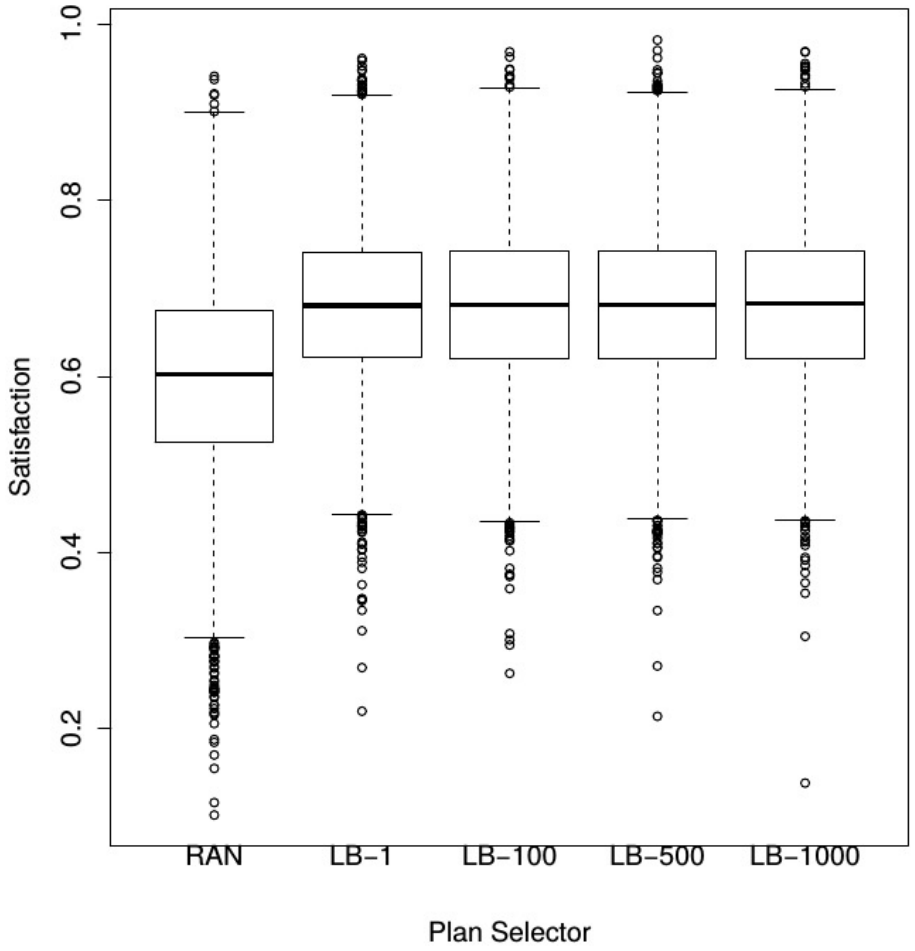
Plan	Softgoal	Influence Factor	Outcome
AirplanePlan	minCosts	AirplaneConditions	FuelConsumption
		WeatherConditions	
		Distance	
	maxPerformance	AirplaneConditions	TimeTaken
		WeatherConditions	
		AirportConditions	
	maxReliability	AirplaneConditions	LoadIntegrity
		AccidentProbability	
		ChanceOfTheft	
ShipPlan	minCosts	ShipConditions	FuelConsumption
		WeatherConditions	
		Distance	
	maxPerformance	ShipConditions	TimeTaken
		WeatherConditions	
		SeaConditions	
		HarborConditions	
	maxReliability	ShipConditions	LoadIntegrity
		AccidentProbability	
		ChanceOfTheft	
TrainPlan	minCosts	TrainConditions	FuelConsumption
		WeatherConditions	
		Distance	
	maxPerformance	TrainConditions	TimeTaken
		TrafficConditions	
		RailroadConditions	
	maxReliability	TrainConditions	LoadIntegrity
		AccidentProbability	
		ChanceOfTheft	
TruckPlan	minCosts	TruckConditions	FuelConsumption
		TrafficConditions	
		Distance	
	maxPerformance	TruckConditions	TimeTaken
		TrafficConditions	
		RoadConditions	
	maxReliability	TruckConditions	LoadIntegrity
		AccidentProbability	
		ChanceOfTheft	

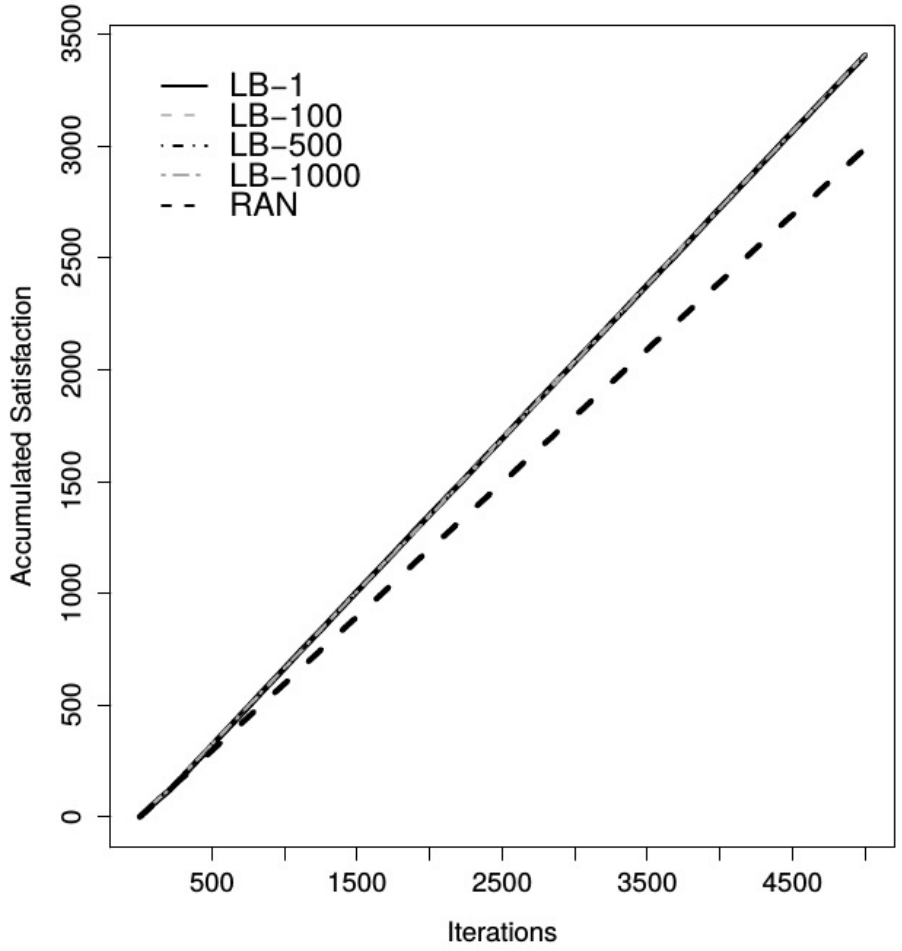
Learning-based Plan Selection

- Measurement of the **agent satisfaction** produced using different plan selection strategies
  - Random-based (**RAN**), LB-**1**, LB-**100**, LB-**500** and LB-**1000**
- Threshold of plan executions: **50**
- Number of iterations: **5000**

Plan Selector	M	SD	Min	Max	Cum Sat
LB-1	0.6812	0.091	0.219	0.961	3406.04
LB-100	0.6815	0.091	**0.262**	0.968	3407.99
LB-500	0.680	0.092	0.214	**0.982**	3402.68
LB-1000	**0.6816**	*0.090*	0.138	0.969	**3408.00**
RAN	*0.597*	**0.114**	*0.102*	*0.941*	*2988.98*

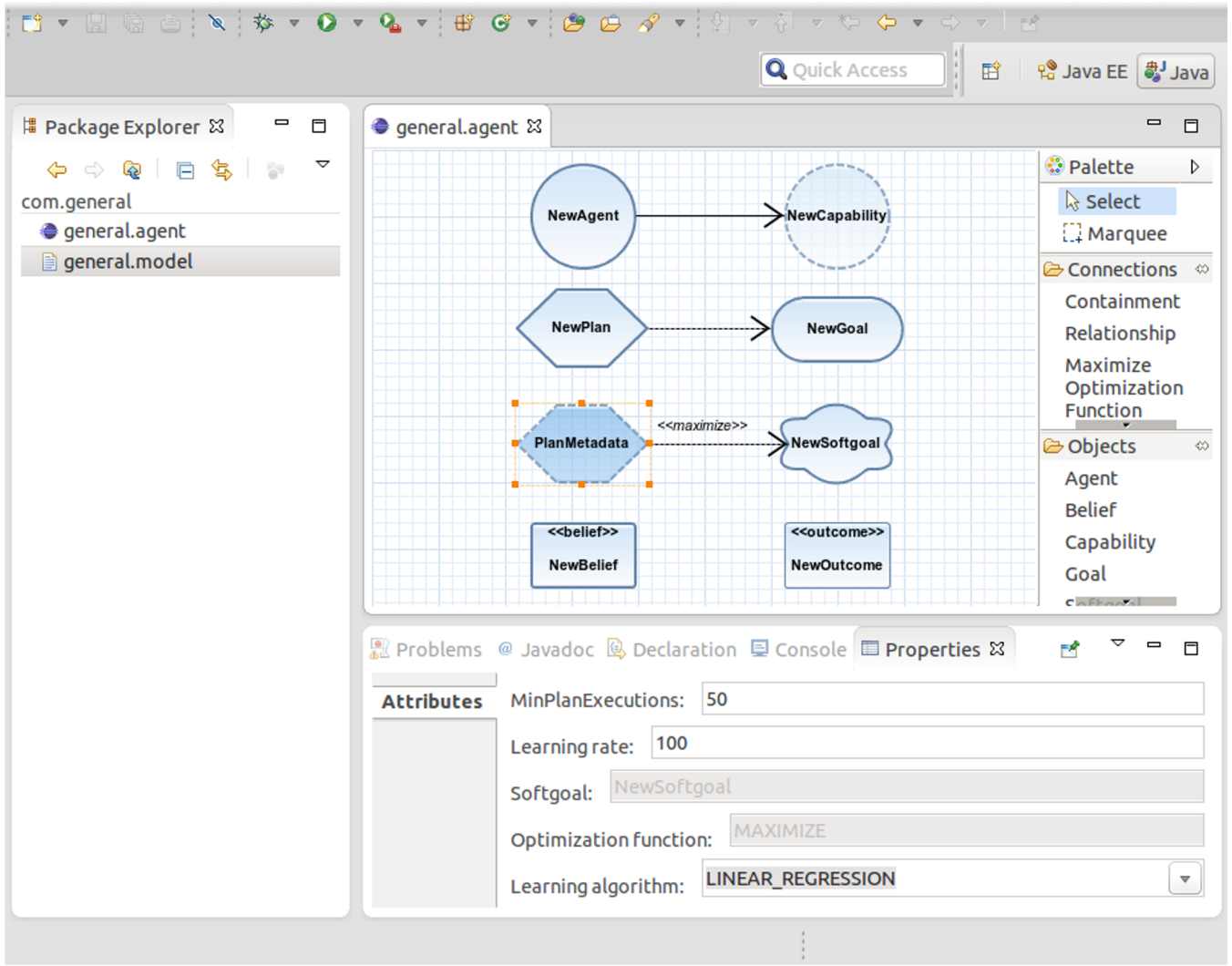
Learning-based Plan Selection

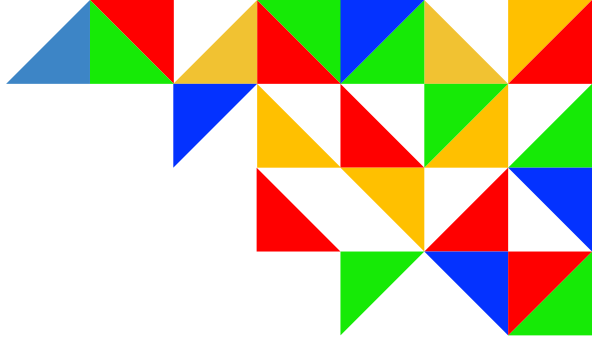




The Sam Tool

- Tool to support the design and implementation of software agents with learning capabilities
- Implemented as a plug-in for Eclipse
- Technologies
  - Graphiti – graphical modelling
  - Xpand – code generation

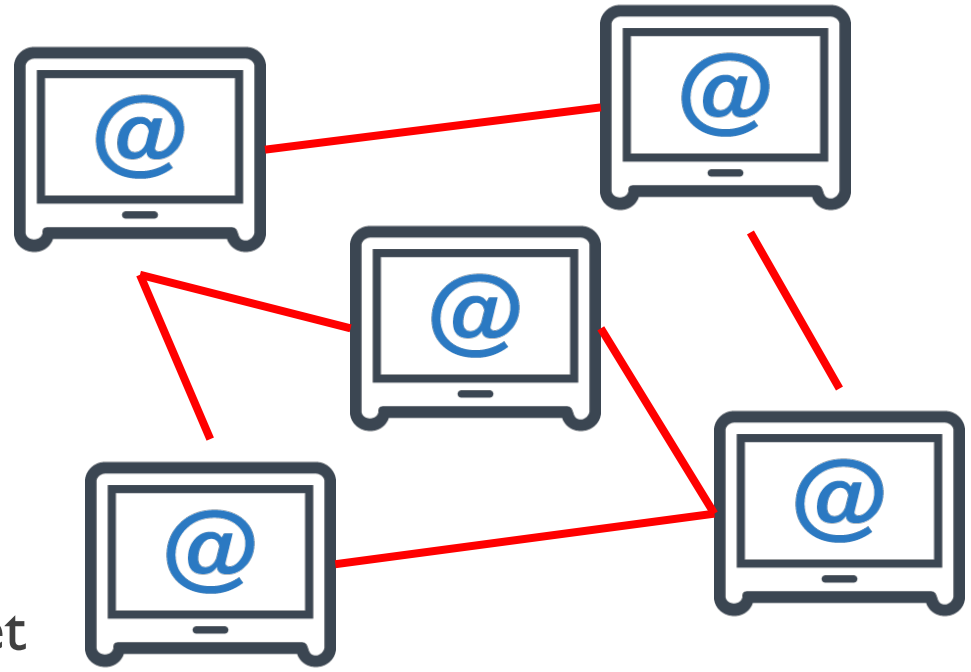




Automated Management of Remediation Actions

Extensions of the BDI Architecture as Reusable Solutions to Ease the Development of Sophisticated BDI Agents

From standalone to connected pieces of software



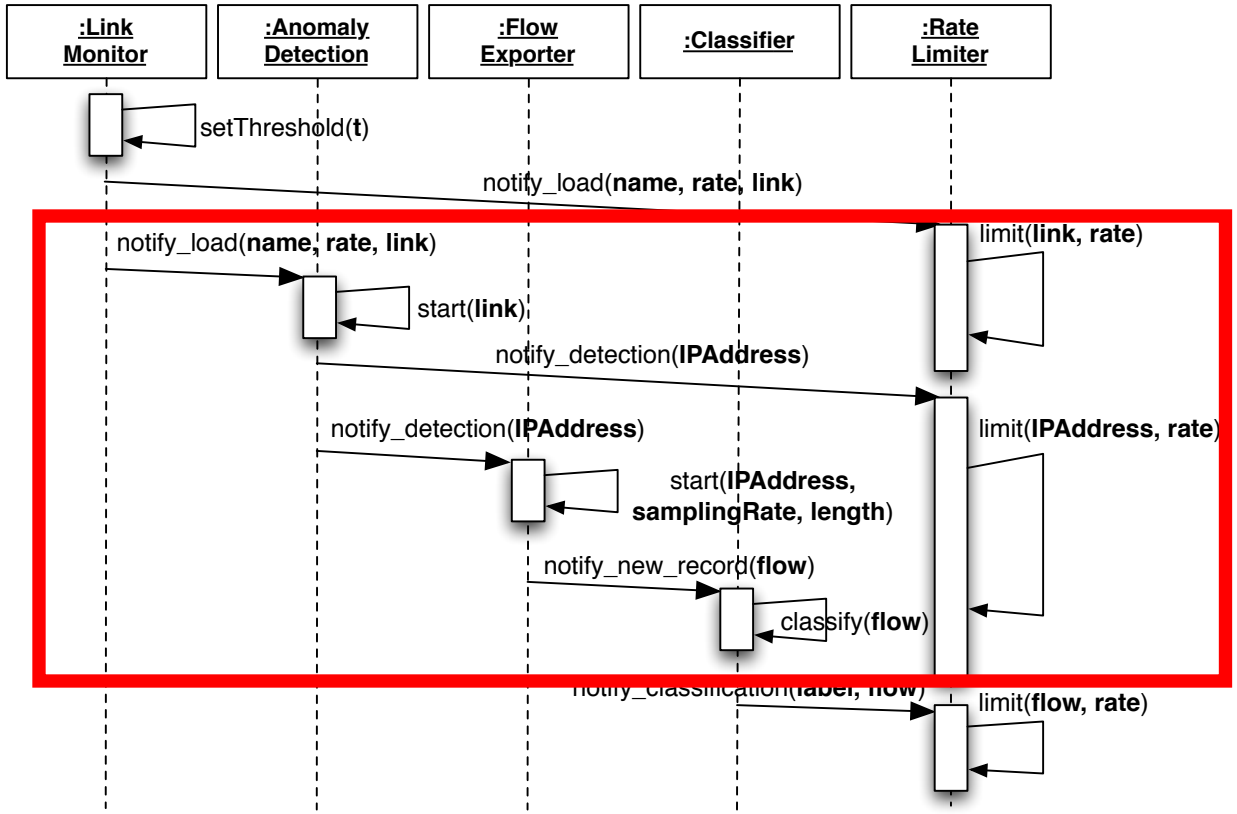
- Software components depend on one another
- Key differences from the past
  - Dynamic environment
  - Little can be assumed from the behaviour of other components
  - Intensive communication based on the internet

RESILIENCE

Automated Management of Remediation Actions

- Remediation Action
  - An action that mitigates the consequences/effects of a problem
- Why?
  - Causes of the problem are unknown
  - Addressing the causes takes too long

- Example

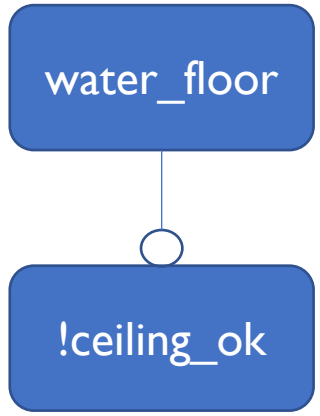


Automated Management of Remediation Actions

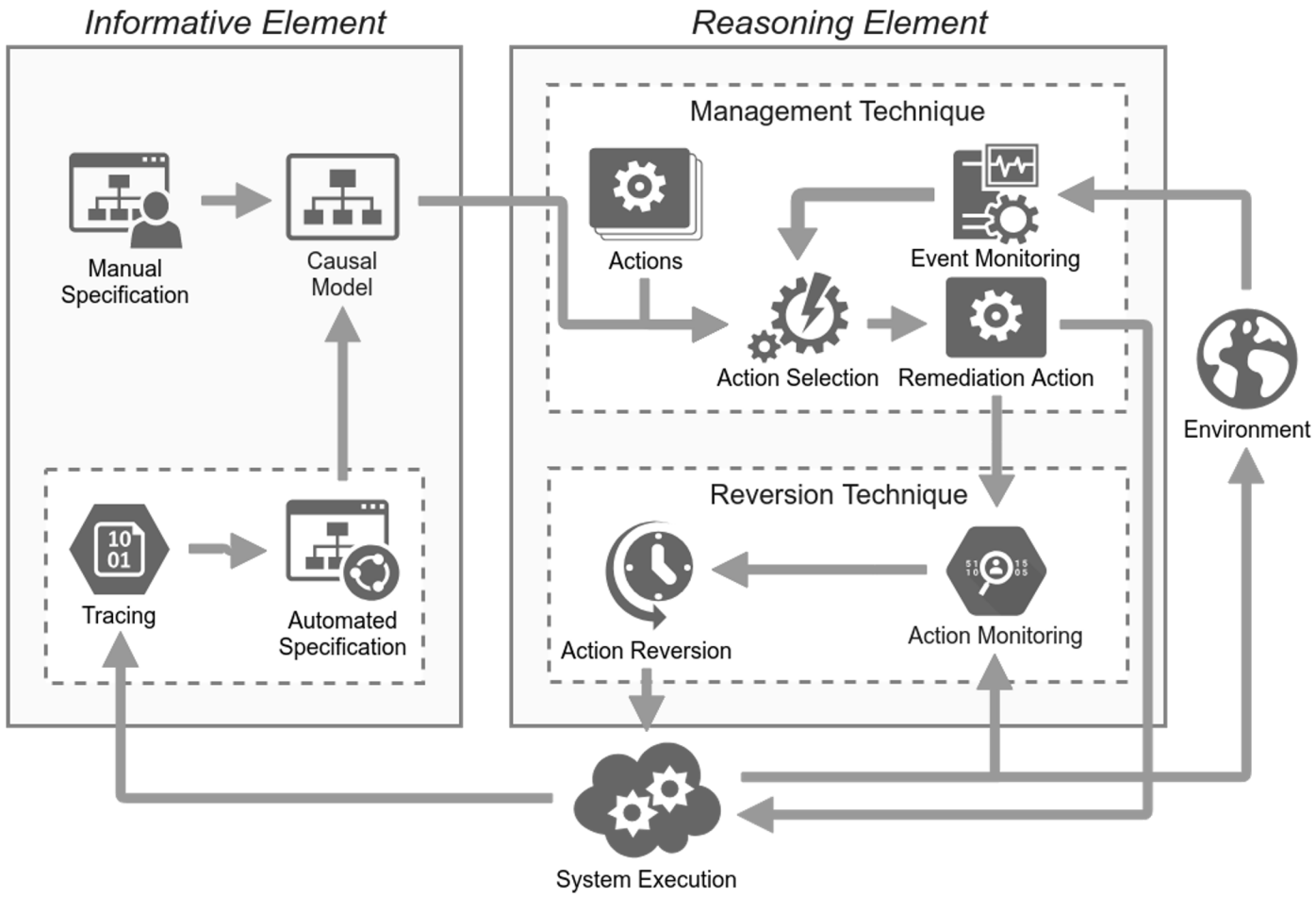
How to **automate** in a **domain-independent way** the management of **remediation actions** and long-term problem resolution so that systems with remedial behaviour can **remain operational** and **recover from abnormal situations** without compromising their performance?

Automated Management of Remediation Actions

- Goal
  - ~water_floor
  - min time
- Plans
  - place_bucket
    - Post-condition: ~water_floor
    - Resources: 1min
  - repair_ceiling
    - Pre-condition: have_tools
    - Post-conditions:
      - ~water_floor
      - ceiling_ok
    - Resources: 60min

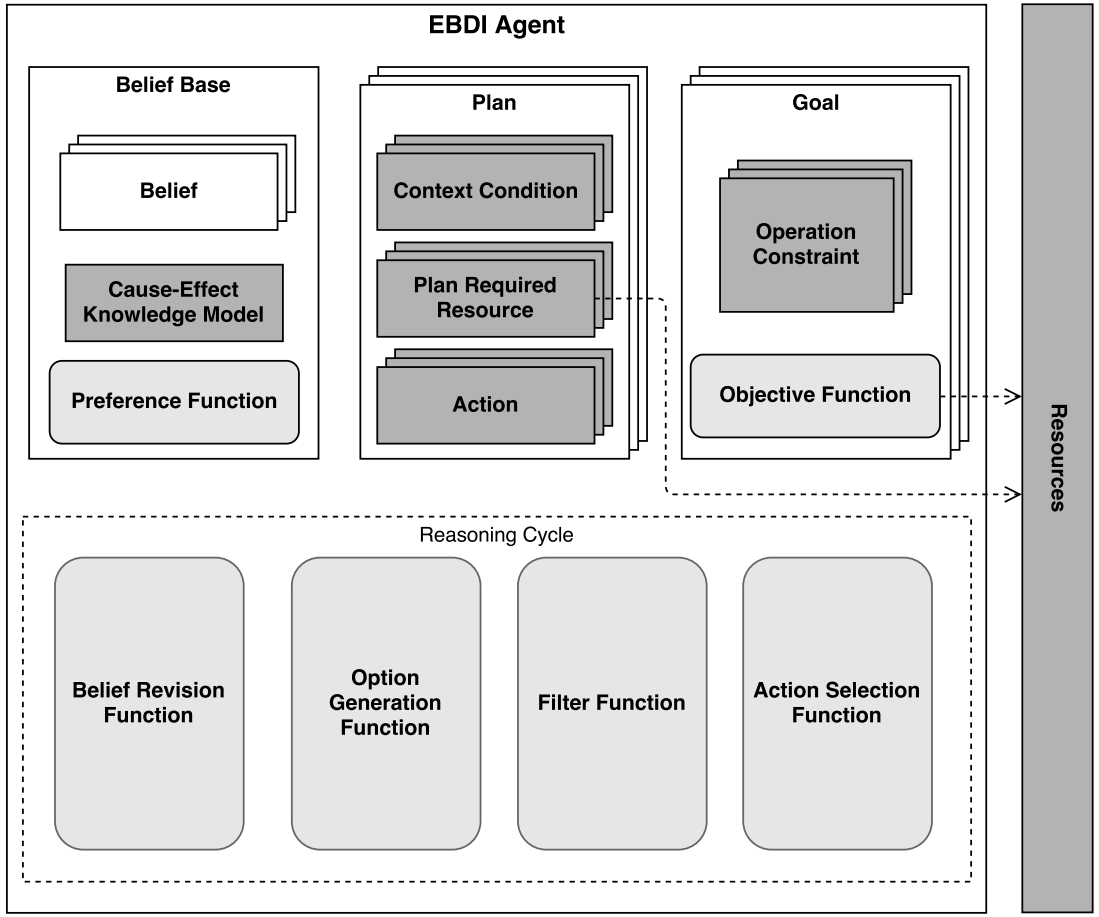


Automated Management of Remediation Actions

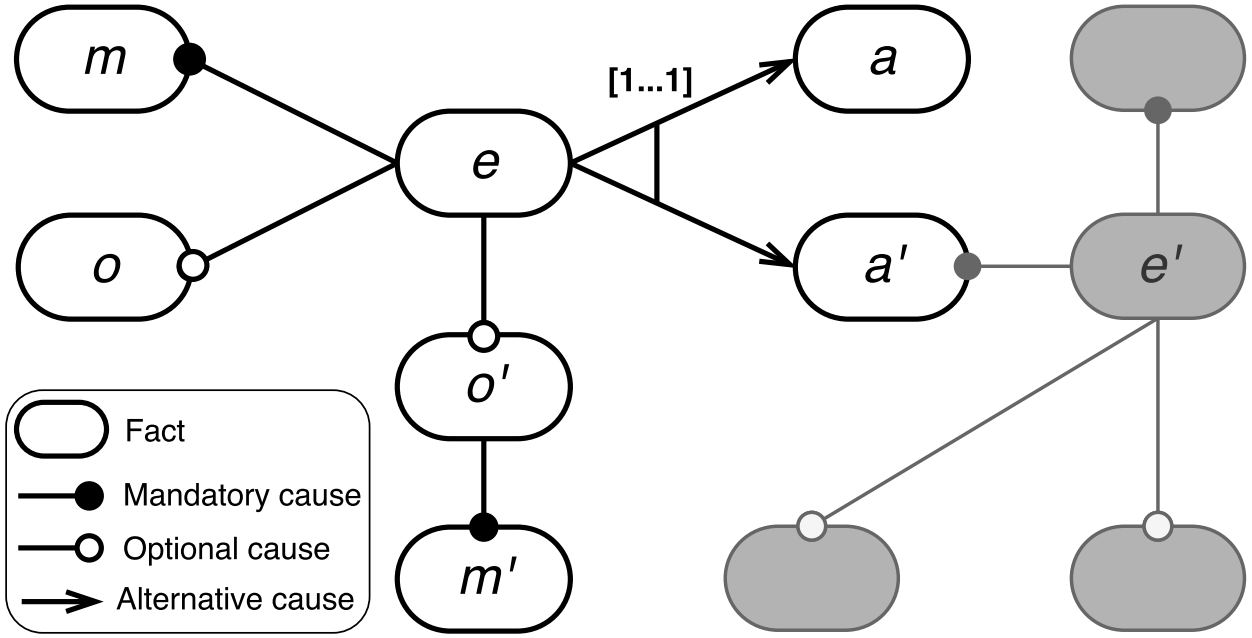


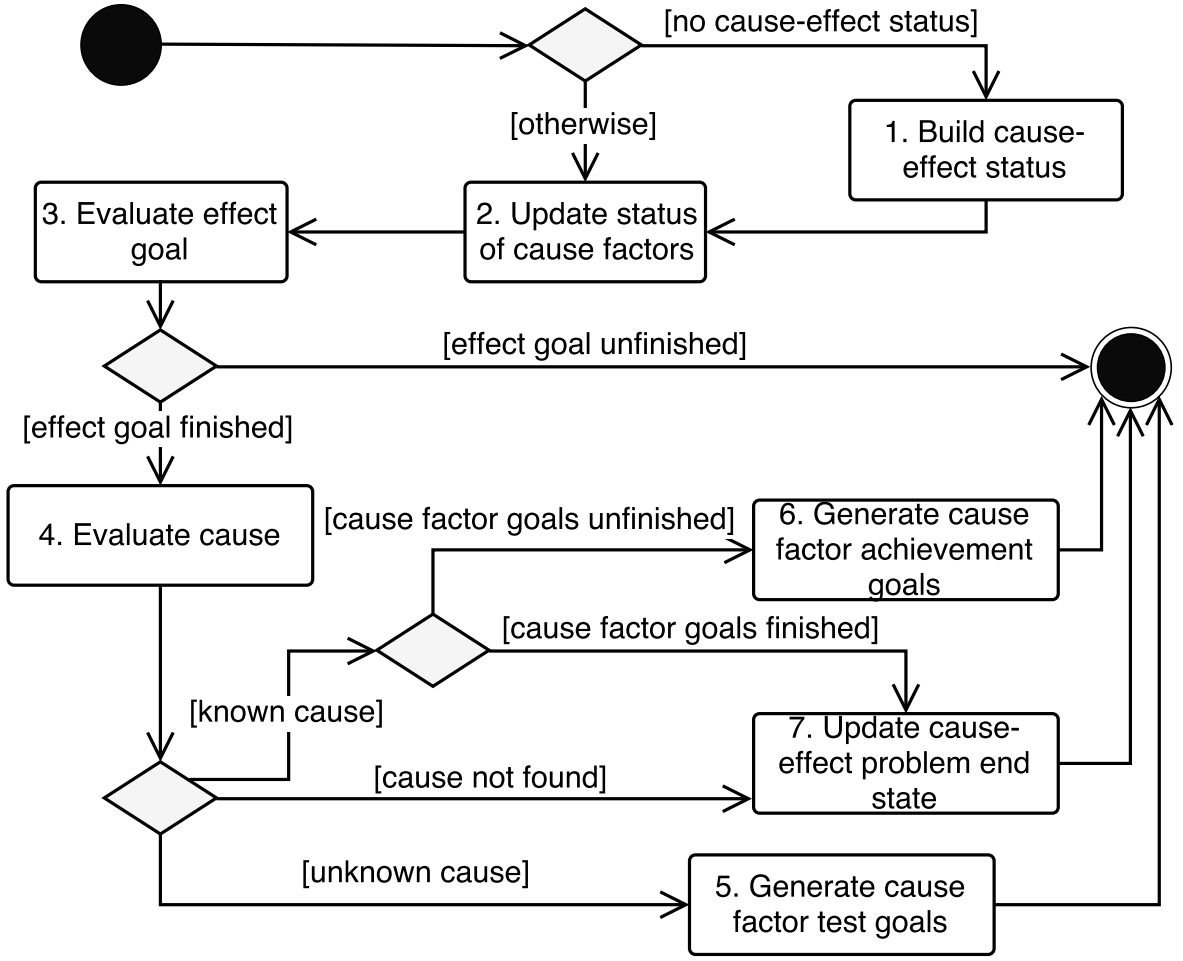
Automated Management of Remediation Actions

1. Constrained Goals/Plan Selection
   - Goal metadata to indicate restrictions of how goals can be achieved
     - constraints + utility-function
2. Cause-effect Knowledge Model
   - Used to identify possible problem causes
3. Goal Generation
   - Identify problem causes
   - Address identified causes

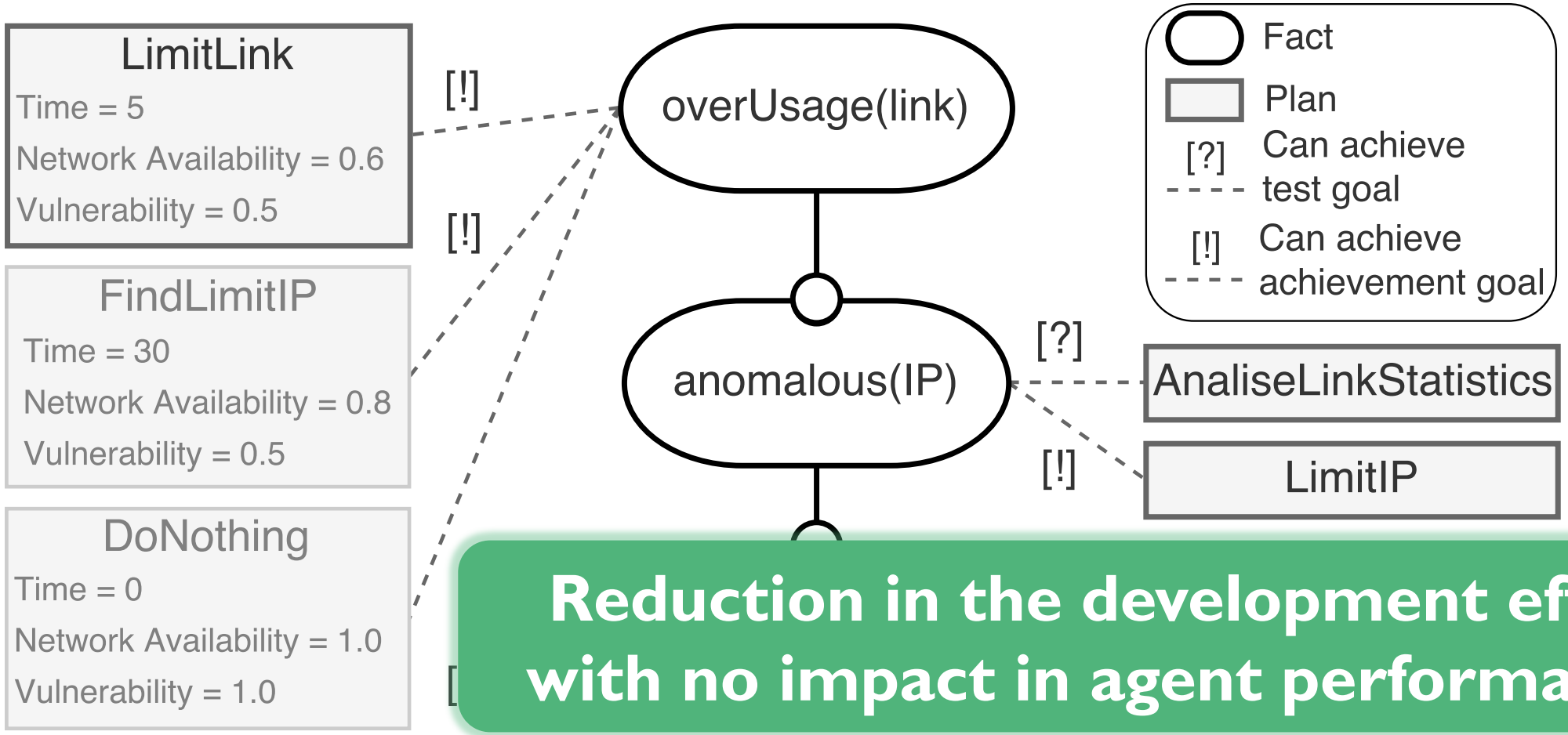


Automated Management of Remediation Actions





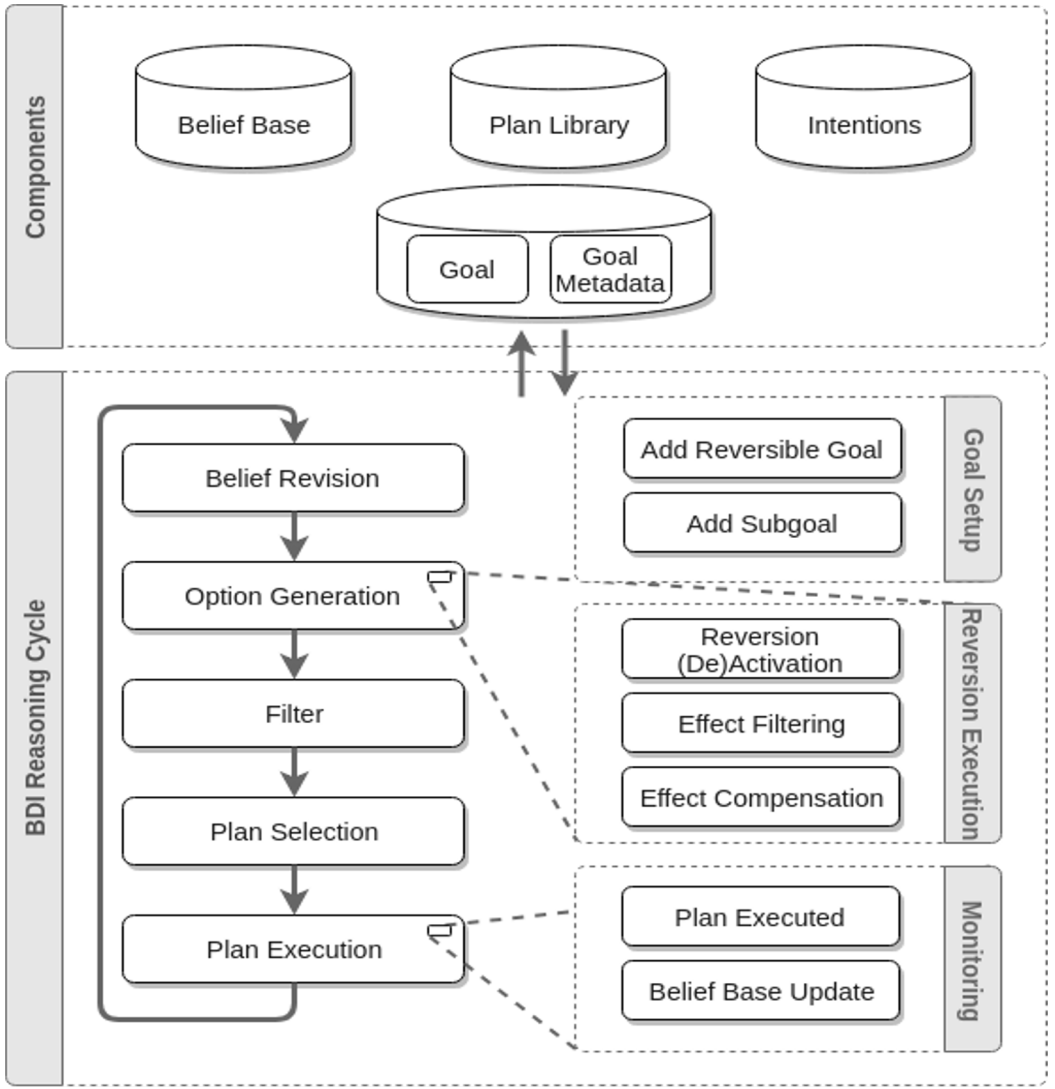
Automated Management of Remediation Actions



Reduction in the development effort with no impact in agent performance

Automated Management of Remediation Actions

- Reversion technique
  - Extends the BDI architecture
  - Specifies steps to be performed by the reasoning cycle
    - Goal Setup
    - Monitoring
    - Reversion Execution



Collaborations and References

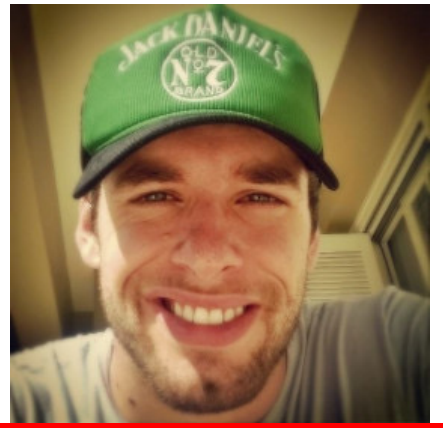
- NUNES, I.; SCHARDONG, F.; SCHAEFFER-FILHO, A. . BDI2DoS: an Application using Collaborating BDI Agents to Combat DDoS Attacks. Journal of Network and Computer Applications, v. 84, p. 14-24, 2017.
- SCHARDONG, F. ; NUNES, I. ; SCHAEFFER FILHO, A.E. A Distributed NFV Orchestrator based on BDI Reasoning. IM 2017.
- SCHARDONG, F. ; NUNES, I. ; SCHAEFFER FILHO, A.E. Providing Cognitive Components with a Bidding Heuristic for Emergent NFV Orchestration. NOMS 2018.

Frederico Schardong

Alberto Schaeffer-Filho

Collaborations and References

- NUNES, I.; LUCK, M. Softgoal-based Plan Selection in Model-driven BDI Agents, AAMAS 2014.
- FACCIN, J. G. ; NUNES, I. BDI-Agent Plan Selection based on Prediction of Plan Outcomes. IAT 2015.
- FACCIN, J. G. ; WENG, J. ; NUNES, I. . SAM: A Tool to Ease the Development of Intelligent Agents. CBSoft 2016 – Tool Session.
- FACCIN, J. G. ; NUNES, I. A tool-supported development method for improved BDI plan selection. ENGINEERING APPLICATIONS OF ARTIFICIAL INTELLIGENCE, v. 62, p. 195-213, 2017.

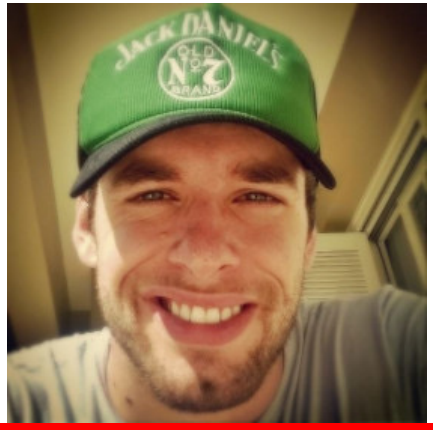


João Guilherme Faccin

Michael Luck

Collaborations and References

- FACCIN, J. ; NUNES, I. Remediating critical cause-effect situations with an extended BDI architecture. EXPERT SYSTEMS WITH APPLICATIONS, v. 95, p. 190-200, 2018.
- FACCIN, J. ; NUNES, I. Cleaning Up the Mess: a Formal Framework for Autonomously Reverting BDI Agent Actions. SEAMS 2018.
- NUNES, I. Improving the Design and Modularity of BDI Agents with Capability Relationships. EMAS 2014.
- NUNES, I.; FACCIN, J. G. . Modelling and Implementing Modularised BDI Agents with Capability Relationships. International Journal of Agent-Oriented Software Engineering, v. 5(2/3), p. 203-231, 2016.



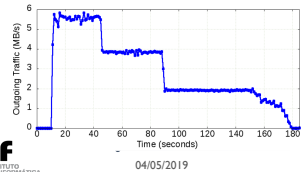
João Guilherme Faccin

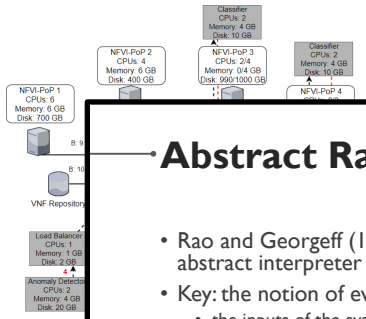
Grupo Prosoft @ UFRGS

Conclusion

BDI Agents to Combat a DDoS Attack

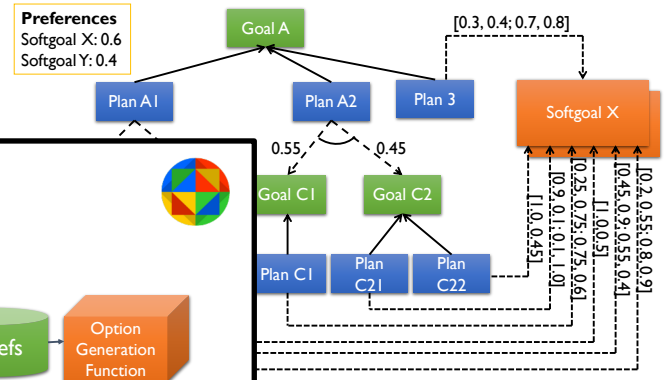
- 1: linkRateLimited(link)
- 2: ?anomalousUsage(link)
- 3: ipRateLimited(ip)
- 4: splitTrafficTowardsIp(ip)
- 5a: flowRecord(ip)
- 5b: flowRecord(ip)
- 6 & 7: flowRateLimited(flow) x 8





04/05/2019 Ingrid Nunes <ingridnunes@in

Plan Selection based on MAUT



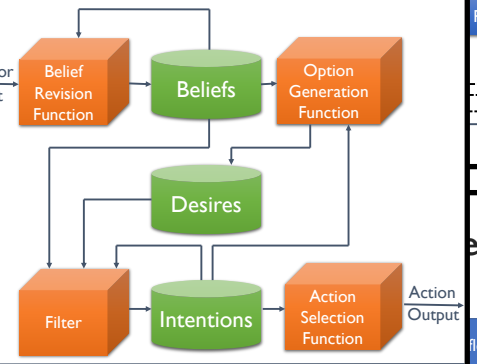
Ingrid Nunes <ingridnunes@inf.ufrgs.br> 50

Abstract Rational Architecture

- Rao and Georgeff (1992) proposed an abstract interpreter for agents
- Key: the notion of events
  - the inputs of the system
  - both externals (from the environment) and internal (within the system)

Algorithm 1 BDI Rational Agent Interpreter

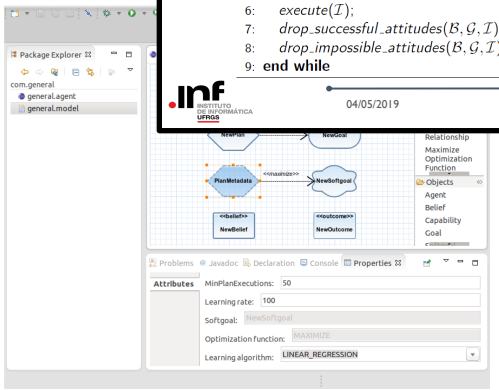
1: *initialize_state*()
2: **while** ¬*quit* **do**
3: *options* ⇐ *option_generator*(*event_queue*, $\mathcal{B}, \mathcal{G}, \mathcal{I}$);
4: *selected_options* ⇐ *deliberate*(*options*, $\mathcal{B}, \mathcal{G}, \mathcal{I}$);
5: *update_intentions*(*selected_options*, $\mathcal{I}$);
6: *execute*($\mathcal{I}$);
7: *drop_successful_attitudes*($\mathcal{B}, \mathcal{G}, \mathcal{I}$);
8: *drop_impossible_attitudes*($\mathcal{B}, \mathcal{G}, \mathcal{I}$);
9: **end while**



04/05/2019 Ingrid Nunes <ingridnunes@inf.ufrgs.br> 3

The Sam Tool

- Tool to support the design and implementation of software agents with learning capabilities
- Implemented as a plug-in for Eclipse
- Technologies
  - Graphiti – graphical modelling
  - Xpand – code generation



04/05/2019 Ingrid Nunes <ingridnunes@inf.ufrgs.br> 70

ement of Remediation

- Plans
  - place_bucket
    - Post-condition: ~water_floor
    - Resources: 1min
  - repair_ceiling
    - Pre-condition: have_tools
    - Post-conditions:
      - ~water_floor
      - ceiling_ok
    - Resources: 60min

04/05/2019 Ingrid Nunes <ingridnunes@inf.ufrgs.br> 75

Abstract

- The Belief-Desire-Intention (BDI) architecture is widely known and largely used to develop autonomous agents. It has key components connected by a reasoning cycle that drives agent actions. This reasoning cycle includes functions that can be customised to provide agents with sophisticated behaviour such as making informed decisions about which course of actions (i.e., a plan) is more suitable considering the current environment state when multiple plans can be used. In this talk, I'll first demonstrate the potential of the BDI architecture by introducing our BDI-agent-based solution for combating DDoS attacks. Then, I'll overview our proposed approaches that extend the BDI architecture to support the development of BDI agents with improved plan selection. Finally, I'll present our latest extension to the BDI architecture that supports the construction of resilient systems by means of BDI agents that can autonomously manage remediation actions with the goal of keeping systems operational under abnormal conditions.