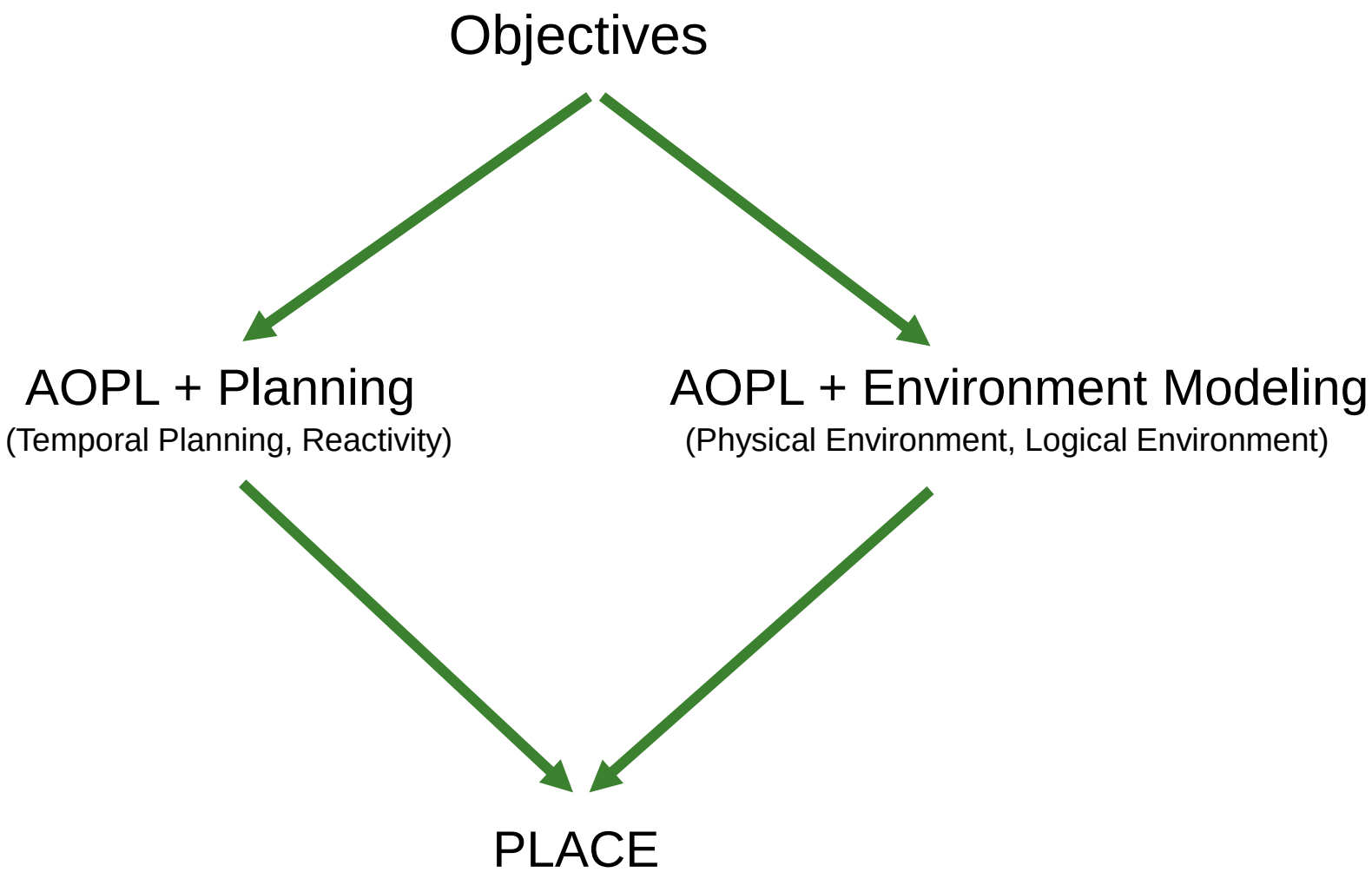
PLACE:
Planning based Language for Agents and Computational Environments

Muhammad Adnan Hashmi
Muhammad.Adnan@cs.uol.edu.pk

Muhammad Usman Akram
musmanakram@ciitlahore.edu.pk

Amal El Fallah Seghrouchni
Amal.ElFallah@lip6.fr



Planning based Language for Agents & Computational Environments

Outline

- Context and Objectives
- Related Work
  - AOP Languages supporting planning
  - EOP Languages
- PLACE
  - Syntax
  - Environment Modeling
  - Planning Mechanism
  - Implementation And GUI
- Case Study
- Conclusion and Perspectives

Outline

- Context and Objectives
- Related Work
  - AOP Languages supporting planning
  - EOP Languages
- PLACE
  - Syntax
  - Environment Modeling
  - Planning Mechanism
  - Implementation And GUI
- Case Study
- Conclusion and Perspectives

Objectives (Environment Modeling)

- Current AOP languages
  - Most AOP languages do not explicitly represent the agent's environment.
    - It is implicitly represented inside the knowledge of agent
    - PROBLEM: Different agents can represent and see the environment differently
  - A framework (A & A) [Ricci et al. 2011] has been proposed but it does not support visual environment modeling
    - PROBLEM: Designer cannot perceive the changes in environment

Objectives (Environment Modeling)

- Current AOP languages
  - Most AOP languages do not explicitly represent the agent's environment.
    - It is implicitly represented inside the knowledge of agent
    - PROBLEM: Different agents can represent and see the environment differently
  - A framework (A & A) [Ricci et al. 2011] has been proposed but it does not support visual environment modeling
    - PROBLEM: Designer cannot perceive the changes in environment
- Our Aim:
  - Propose an AOP language that supports:
    - Explicit environment representation
    - Visual environment modeling

Objectives (Reasoning Mechanism)

- Current languages
  - Most AOP languages follow a reactive (PRS based) approach
  - AOP languages do not support temporal planning
    - Only a few support planning
- Problems
  - Execution without planning may result in the goal failures
    - Agent can reach a dead end
  - Conflicts can arise among different plans
  - Real world actions take place over a timespan

Objectives (Reasoning Mechanism)

- Current languages
  - Most AOP languages follow a reactive (PRS based) approach
  - AOP languages do not support temporal planning
    - Only a few support planning
- Problems
  - Execution without planning may result in the goal failures
    - Agent can reach a dead end
  - Conflicts can arise among different plans
  - Real world actions take place over a timespan
- Our Aim:
  - Propose an AOP language that supports:
    - Temporal Planning
    - Goals of different priorities

Our Contribution

- PLACE
  - Has temporal planning
  - Handles *on the fly* goals having different priorities
  - Supports environment modeling and representation

Outline

- Context and Objectives
- Related Work
  - AOP Languages supporting planning
  - EOP Languages
- PLACE
  - Syntax
  - Environment Modeling
  - Planning Mechanism
  - Implementation And GUI
- Case Study
- Conclusion and Perspectives

Related Work (Planning based AOPL)

- CANPLAN [De Silva et al. 2005] combines JSHOP with JACK
  - Programmer's responsibility to call the planner

- X-BDI [Meneguzzi et al. 2004] incorporates classical planning in a BDI framework
  - Efficiency concerns
  - Loss of the domain information

- CYPRESS [Myers et al. 1994] integrates SIPE-2 with PRS

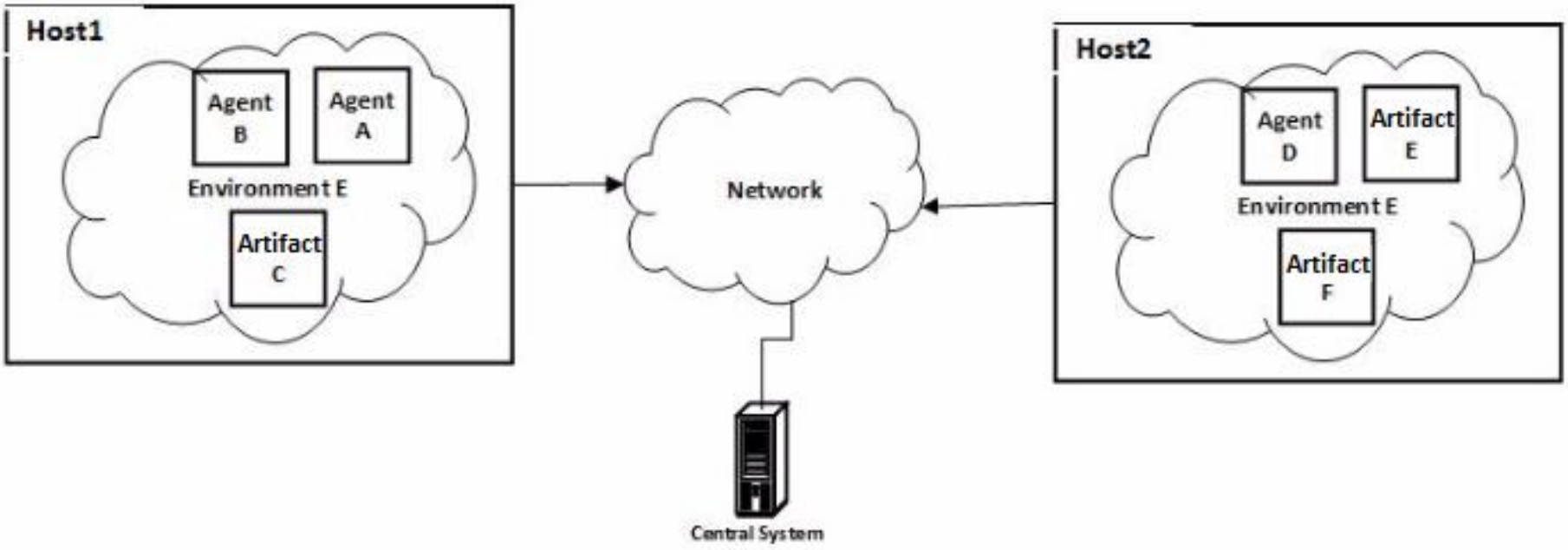
Related Work (EOPL)

- AGRE [J. Ferber et al 2005]
  - (Agent-Group-Role) Organizational model with a notion of environment
- GOLEM [S. Bromuri et al 2008]
  - Environments for situated cognitive agents
  - Do not support mobility of the agents
- CArtAgO [A. Ricci et al 2011]
  - Agent & Artifact (A & A) Model
  - Java APIs for programming the environment

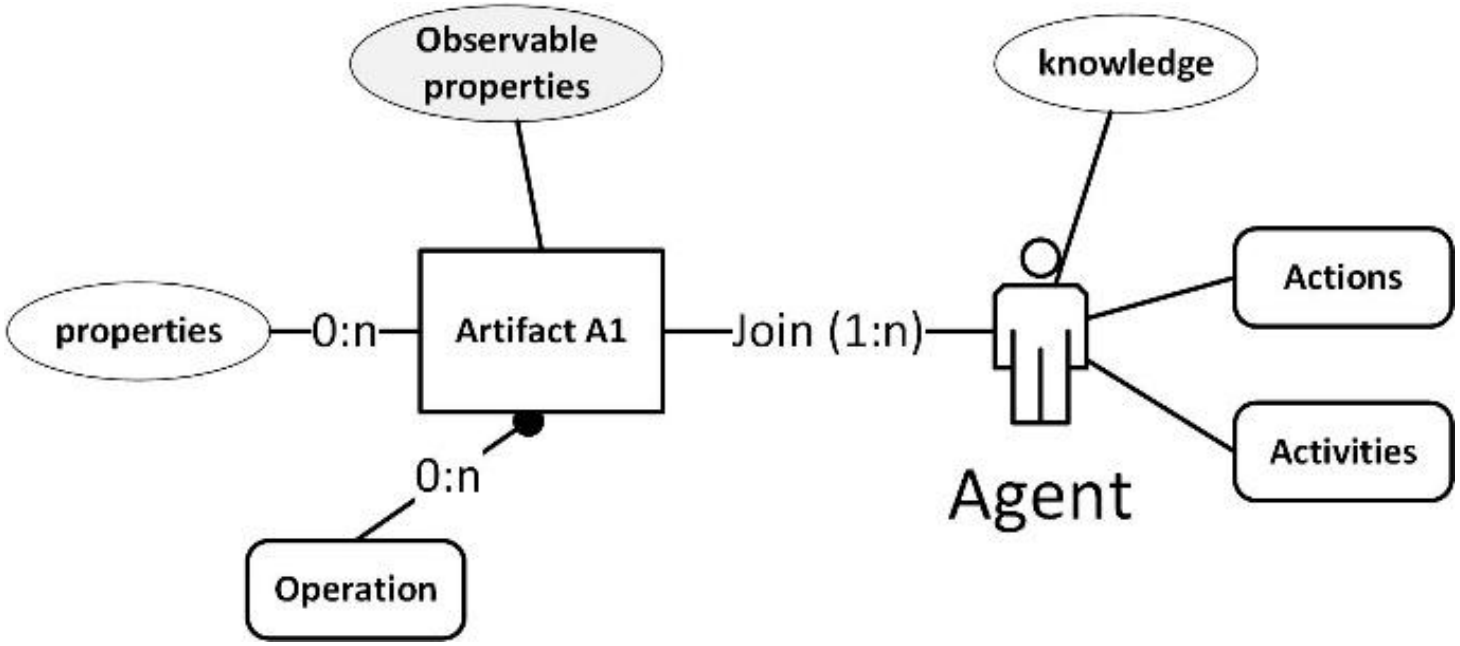
Outline

- Context and Objectives
- Related Work
  - AOP Languages supporting planning
  - EOP Languages
- PLACE
  - Syntax
  - Environment Modeling
  - Planning Mechanism
  - Implementation And GUI
- Case Study
- Conclusion and Perspectives

PLACE Environment Deployment



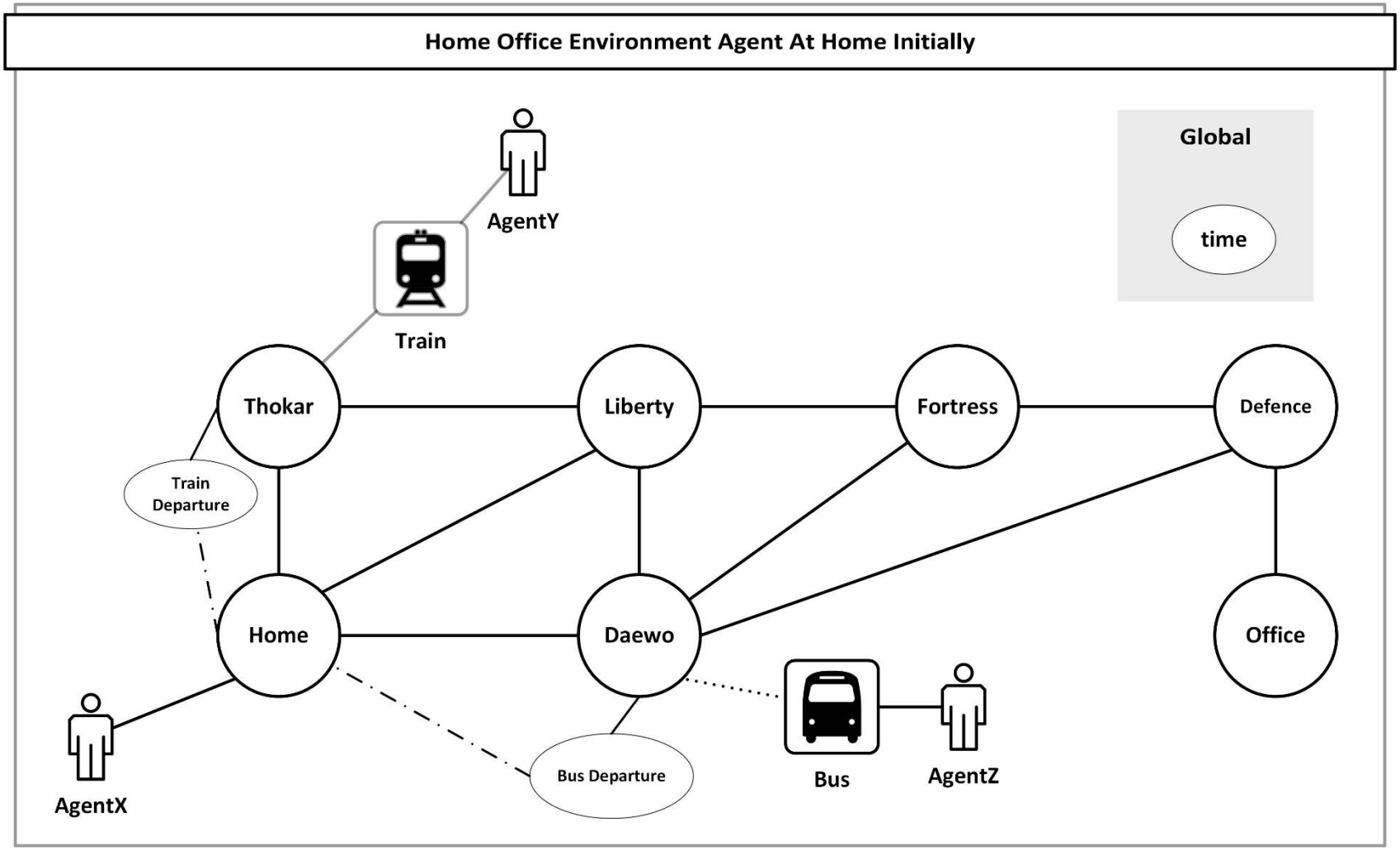
PLACE Environment Meta Model



Outline

- Context and Objectives
- Related Work
  - AOP Languages supporting planning
  - EOP Languages
- PLACE
  - Syntax
  - Environment Modeling
  - Planning Mechanism
  - Implementation And GUI
- Case Study
- Conclusion and Perspectives

A Simple Case Study



PLACE Syntax

- Defining Agents

```
defineAgent agentName {
        knowledge = null; | { (knowledge;)+ }
        goals = null; |{ (goal;)+ }
        actions = null; | { (action)+ }
        activities = null; | { (activity)+ }
        environment = null; | { environmentName }
        agentIn = null; | { artifactName }
        artifacts = null;| { (artifactName;)+ }
}
```

PLACE Syntax

- Defining Agents

defineAgent agentName {

knowledge = null; | { (knowledge;)+ }
goals = null; |{ (goal;)+ }
actions = null; | { (action)+ }
activities = null; | { (activity)+ }
environment = null; | { environmentName }
agentIn = null; | { artifactName }
artifacts = null;| { (artifactName;)+ }

}

Example:

defineAgent AgentX{ /* Comments */}

PLACE Syntax

- Defining Agents

```
defineAgent agentName {
        knowledge = null; | { (knowledge;)+ }
        goals = null; |{ (goal;)+ }
        actions = null; | { (action)+ }
        activities = null; | { (activity)+ }
        environment = null; | { environmentName }
        agentIn = null; | { artifactName }
        artifacts = null;| { (artifactName;)+ }
}
```

Example:

knowledge={connectedTo(Home,Thokar);}

PLACE Syntax

- Defining Agents

```
defineAgent agentName {
        knowledge = null; | { (knowledge;)+ }
        goals = null; |{ (goal;)+ }
        actions = null; | { (action)+ }
        activities = null; | { (activity)+ }
        environment = null; | { environmentName }
        agentIn = null; | { artifactName }
        artifacts = null;| { (artifactName;)+ }
}
```

Example:

goals={agentIn(this, Office);}

PLACE Syntax

- Defining Agents

```
defineAgent agentName {
        knowledge = null; | { (knowledge;)+ }
        goals = null; |{ (goal;)+ }
        actions = null; | { (action)+ }
        activities = null; | { (activity)+ }
        environment = null; | { environmentName }
        agentIn = null; | { artifactName }
        artifacts = null;| { (artifactName;)+ }
}
```

Example:

```
actions = { boardTrain(?p) {
  preconditions = {and(agentIn(this, ?p)), artifactIn(Train,?p), hasKnowledge(hasTrainTicket()))}
  add_effects = {agentIn(this, Train)}
  del_effects = {agentIn(this, ?p)}
  duration=2;
  } ……..
}
```

PLACE Syntax

- Defining Agents

```
defineAgent agentName {
        knowledge = null; | { (knowledge;)+ }
        goals = null; |{ (goal;)+ }
        actions = null; | { (action)+ }
        activities = null; | { (activity)+ }
        environment = null; | { environmentName }
        agentIn = null; | { artifactName }
        artifacts = null;| { (artifactName;)+ }
}
```

Example:

```
activities={ catchTrain(?p) {
  preconditions = (hasKnowledge(agentIn(Home));
  do{moveTo(?p).boardTrain(?p)}
  }
  ……
}
```

PLACE Syntax

- Defining Agents

```
defineAgent agentName {
        knowledge = null; | { (knowledge;)+ }
        goals = null; |{ (goal;)+ }
        actions = null; | { (action)+ }
        activities = null; | { (activity)+ }
        environment = null; | { environmentName }
        agentIn = null; | { artifactName }
        artifacts = null;| { (artifactName;)+ }
}
```

Example:

environment = {HomeOffice}

PLACE Syntax

- Defining Agents

```
defineAgent agentName {
        knowledge = null; | { (knowledge;)+ }
        goals = null; |{ (goal;)+ }
        actions = null; | { (action)+ }
        activities = null; | { (activity)+ }
        environment = null; | { environmentName }
        agentIn = null; | { artifactName }
        artifacts = null;| { (artifactName;)+ }
}
```

Example:

agentIn = {Thokar}

PLACE Syntax

- Defining Agents

```
defineAgent agentName {
        knowledge = null; | { (knowledge;)+ }
        goals = null; |{ (goal;)+ }
        actions = null; | { (action)+ }
        activities = null; | { (activity)+ }
        environment = null; | { environmentName }
        agentIn = null; | { artifactName }
        artifacts = null;| { (artifactName;)+ }
}
```

Example:

artifacts = {Laptop;Wallet;} //agent's possessions

PLACE Syntax

- Defining Artifacts

```
defineArtifact artifactName {
        parent = null; | {artifactName}
        connectedTo = null; | { (artifactName[{notBreakable}];)+ }
        environment = null; | { environmentName }
        properties = null; | { (property[{notChangeable}][observableTo = {(
        artifactName;)+}];)+ }
        operations = null; | { (operation)+ }
}
```

PLACE Syntax

- Defining Artifacts

defineArtifact artifactName {

parent = null; | {artifactName}
connectedTo = null; | { (artifactName[{notBreakable}];)+ }
environment = null; | { environmentName }
properties = null; | { (property[{notChangeable}][observableTo = {(
artifactName;)+}];)+ }
operations = null; | { (operation)+ }

}

Example:

defineArtifact Train{ /* Comments */}

PLACE Syntax

- Defining Artifacts

```
defineArtifact artifactName {
        parent = null; | {artifactName}
        connectedTo = null; | { (artifactName[{notBreakable}];)+ }
        environment = null; | { environmentName }
        properties = null; | { (property[{notChangeable}][observableTo = {(
        artifactName;)+}];)+ }
        operations = null; | { (operation)+ }
}
```

Example:

parent = { Thokar }

PLACE Syntax

- Defining Artifacts

```
defineArtifact artifactName {
        parent = null; | {artifactName}
        connectedTo = null; | { (artifactName[{notBreakable}];)+ }
        environment = null; | { environmentName }
        properties = null; | { (property[{notChangeable}][observableTo = {(
        artifactName;)+}];)+ }
        operations = null; | { (operation)+ }
}
```

Example:

connectedTo = null;

PLACE Syntax

- Defining Artifacts

```
defineArtifact artifactName {
        parent = null; | {artifactName}
        connectedTo = null; | { (artifactName[{notBreakable}];)+ }
        environment = null; | { environmentName }
        properties = null; | { (property[{notChangeable}][observableTo = {(
        artifactName;)+}];)+ }
        operations = null; | { (operation)+ }
}
```

Example:

environment = { HomeOffice }

PLACE Syntax

- Defining Artifacts

```
defineArtifact artifactName {
        parent = null; | {artifactName}
        connectedTo = null; | { (artifactName[{notBreakable}];)+ }
        environment = null; | { environmentName }
        properties = null; | { (property[{notChangeable}][observableTo = {(
        artifactName;)+}];)+ }
        operations = null; | { (operation)+ }
}
```

Example:

properties = {trainDepartureTime(this, 15){obseravableTo={Home}};}

PLACE Syntax

- Defining Artifacts

```
defineArtifact artifactName {
        parent = null; | {artifactName}
        connectedTo = null; | { (artifactName[{notBreakable}];)+ }
        environment = null; | { environmentName }
        properties = null; | { (property[{notChangeable}][observableTo = {(
        artifactName;)+}];)+ }
        operations = null; | { (operation)+ }
}
```

Example:

```
operations={
  driveTo(?s,?d,?a){
    agents = {AgentY}
    preconditions = {
     and(agentIn(?a,this), artifactIn(this,?s),trainDepartureTime(?s,?t), >=(java.currentTime(),?t))
    }
    add_effects_artifact = {artifactIn(this,?d)}}
  }
}
```

PLACE Syntax

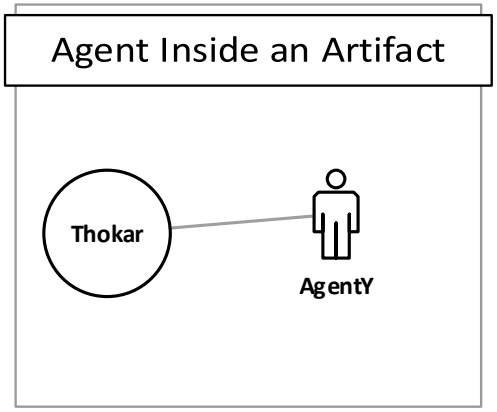
- Defining Environments

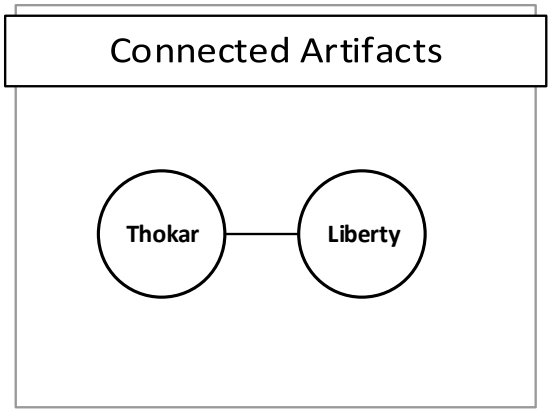
```
defineEnvironment environmentName {
        properties = null; | { (property;)+}
        operations = null; | { (operation)+}
}
```

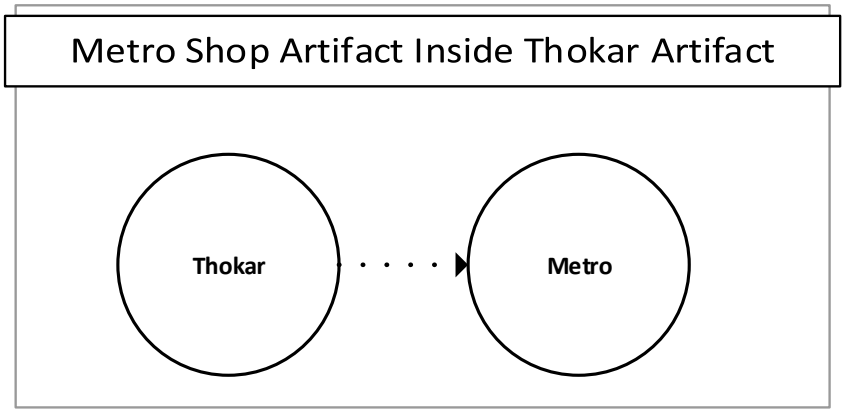
Outline

- Context and Objectives
- Related Work
  - AOP Languages supporting planning
  - EOP Languages
- PLACE
  - Syntax
  - Environment Modeling
  - Planning Mechanism
  - Implementation And GUI
- Case Study
- Conclusion and Perspectives

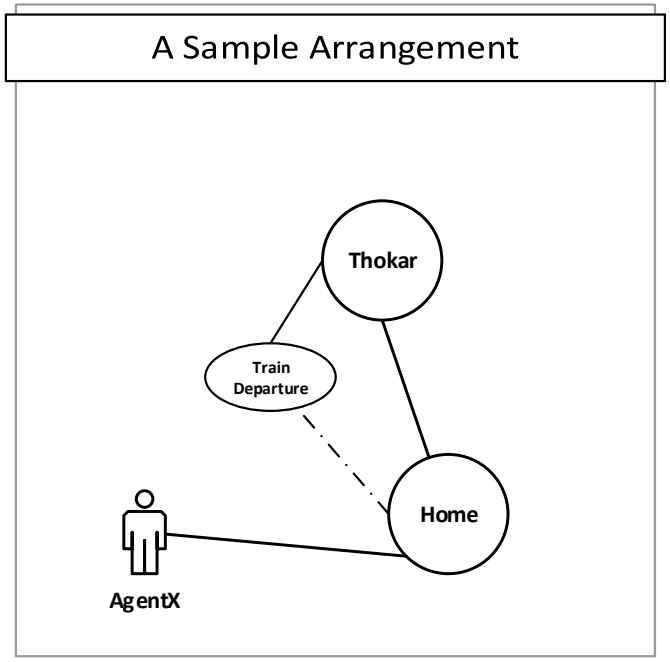
Agent & Artifacts Representation







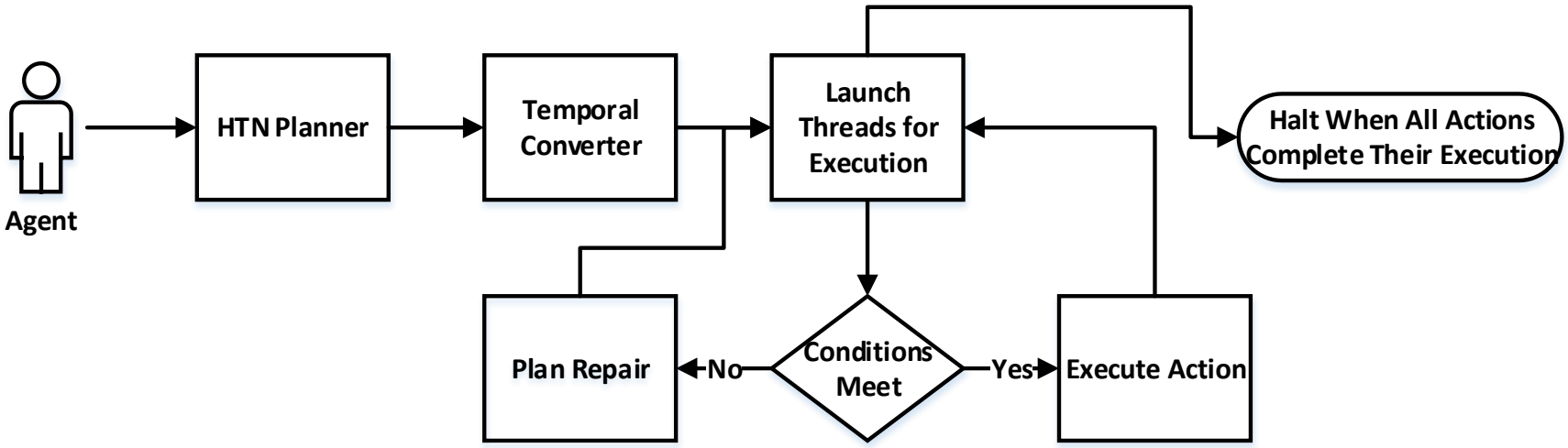
Observable Properties Representation



Outline

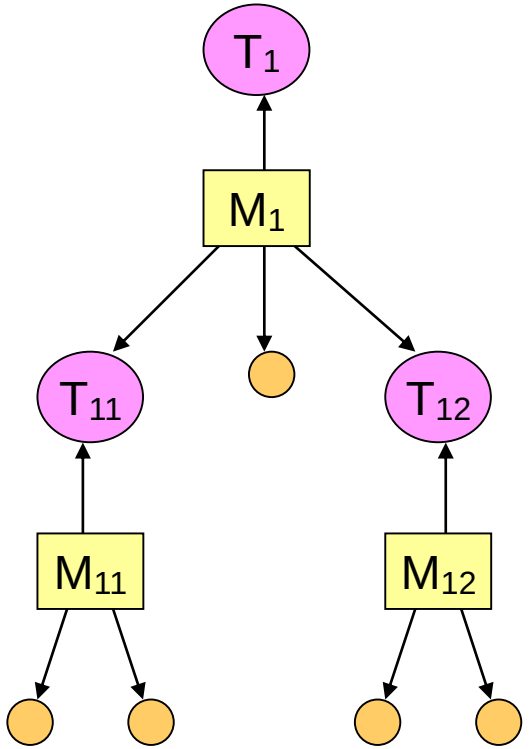
- Context and Objectives
- Related Work
  - AOP Languages supporting planning
  - EOP Languages
- PLACE
  - Syntax
  - Environment Modeling
  - Planning Mechanism
  - Implementation And GUI
- Case Study
- Conclusion and Perspectives

Planner Flow Chart

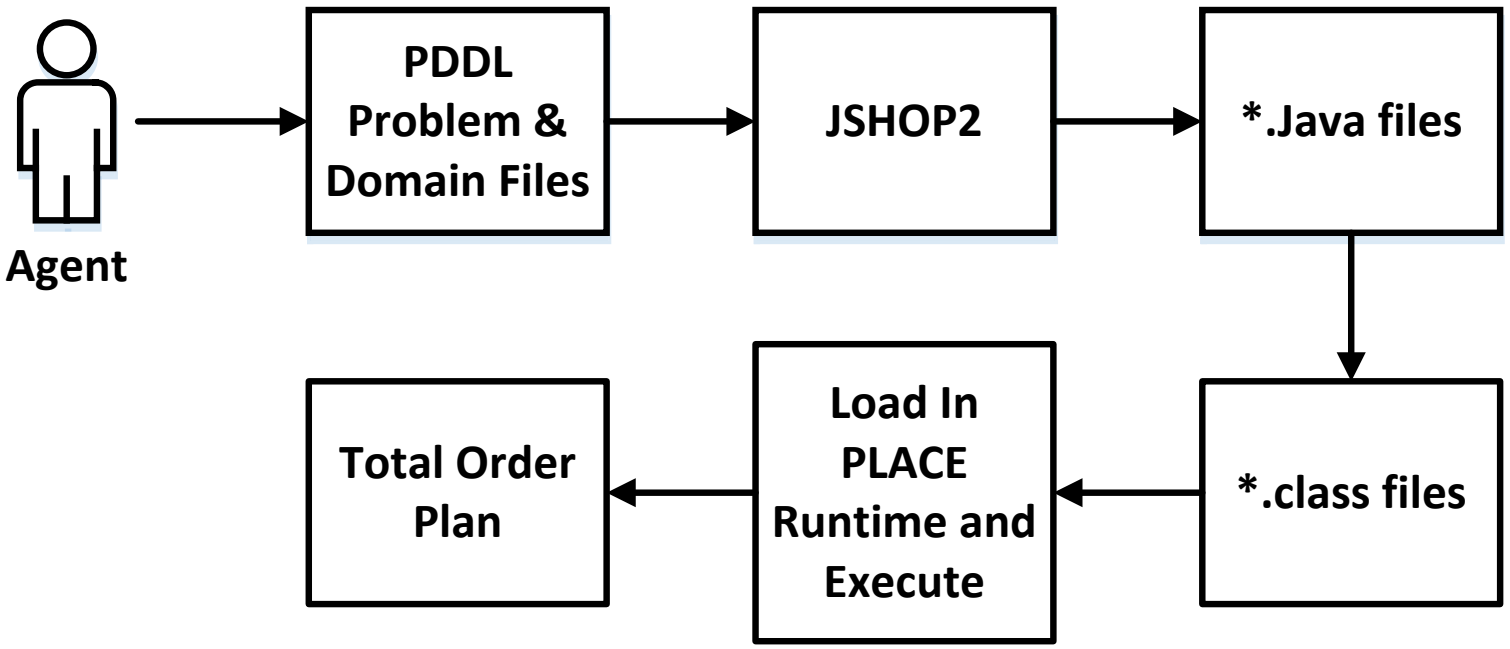


Plan Computation

- JSHOP2 algorithm [Nau et el. 2003] is used to compute a totally ordered plan for each goal
  - An HTN planning algorithm
  - Decomposes the task into sub-tasks by applying methods
  - Recursively applies the same procedure on every composite sub-task until there are only primitive tasks



JSHOP2 in PLACE



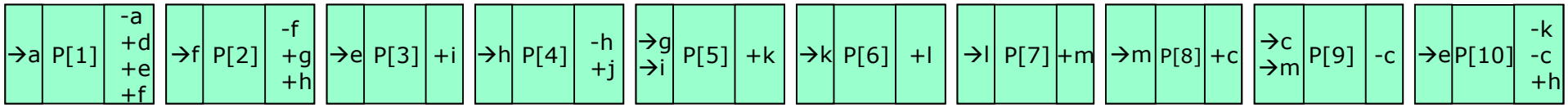
Temporal Converter [Hashmi 2012]

- Input to procedure
  - A totally ordered plan
  - Actions' information (Add, Del, Pre, Durations)

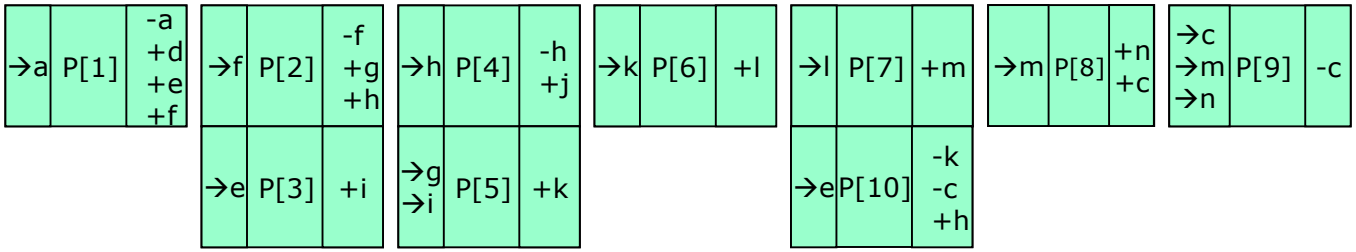
- Output of procedure
  - A position constrained parallel plan
    - Every action is assigned a time stamp
    - Multiple actions can possibly lie in parallel

Temporal Converter (Example)

Input Plan

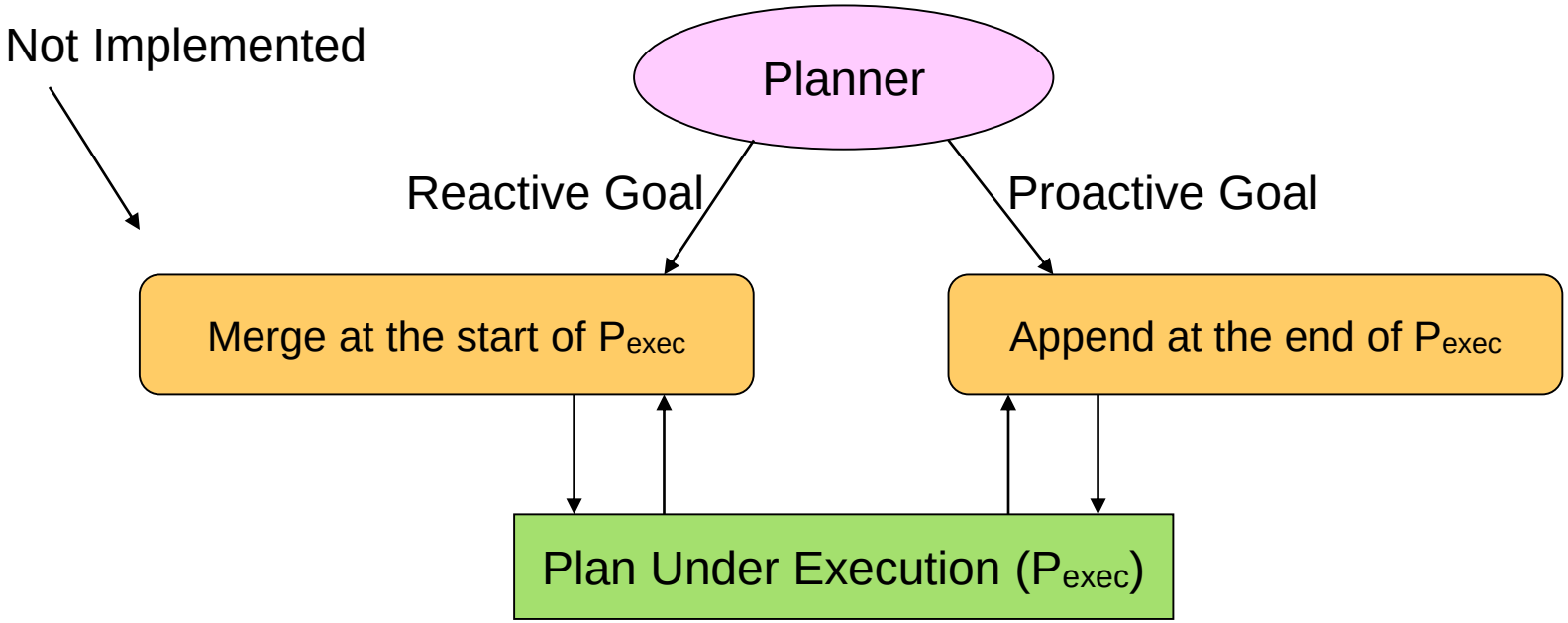


Output Plan



Makespan Gain: 30%

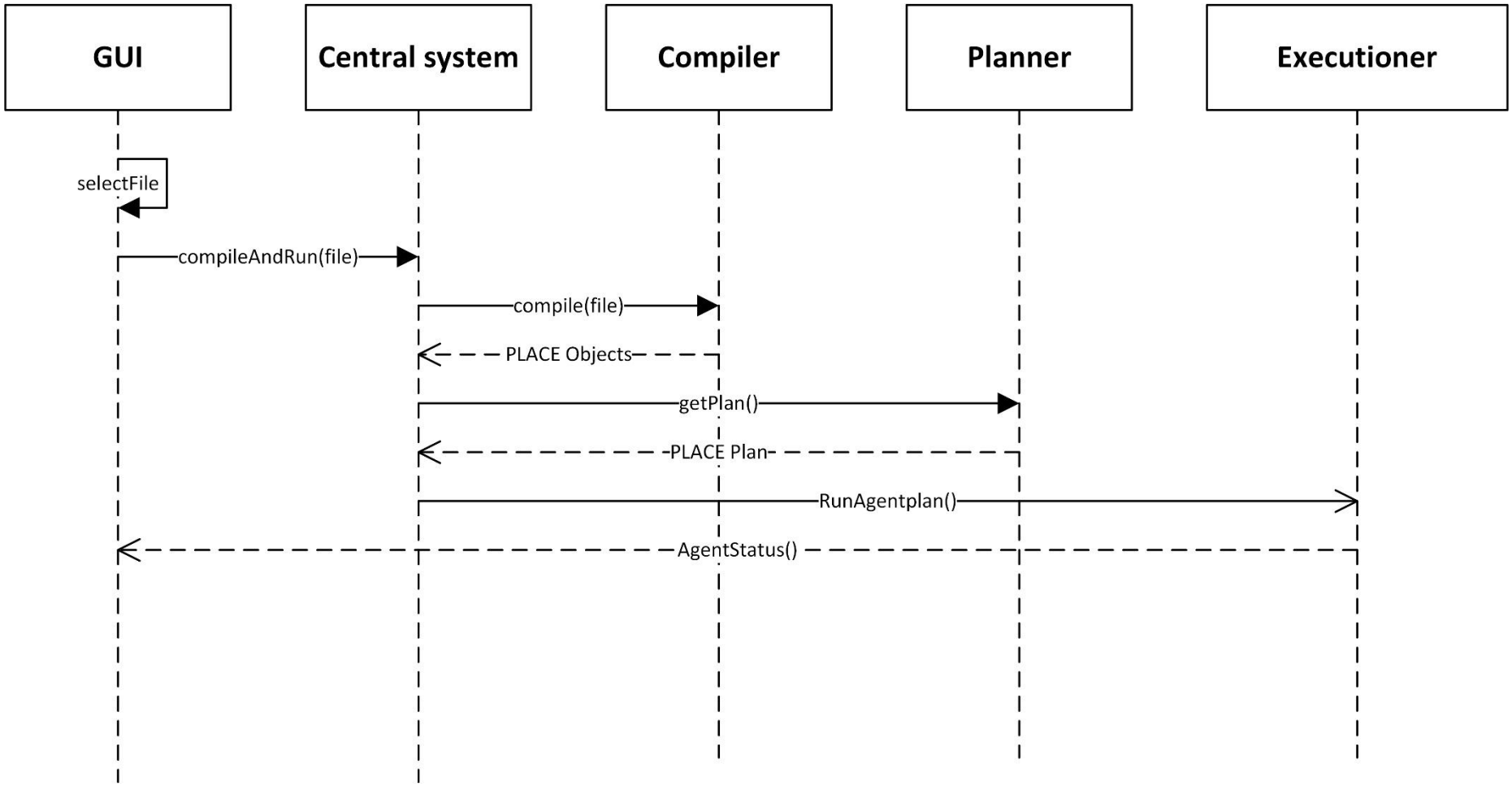
Merging the New Plan to Global Plan



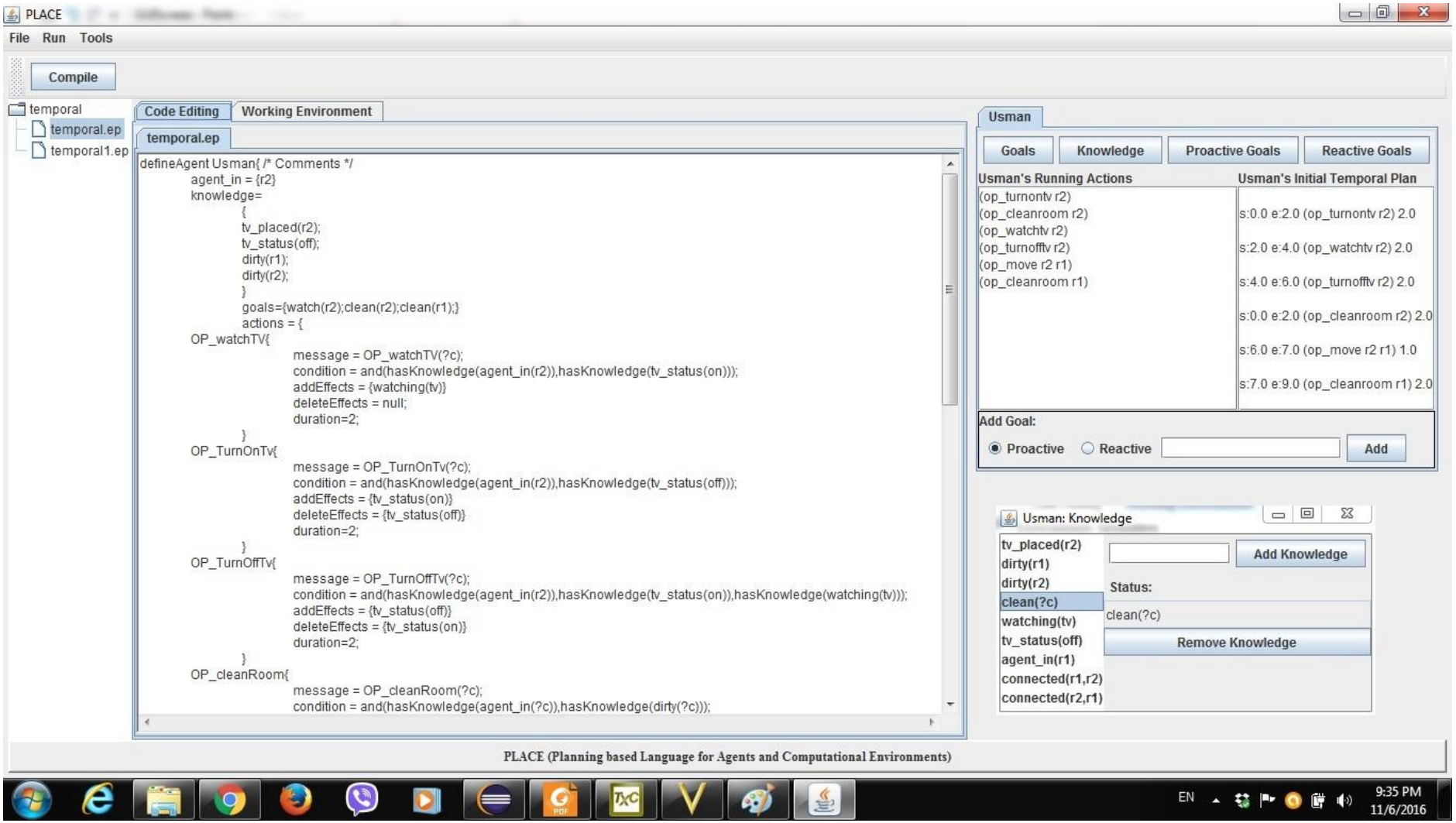
Outline

- Context and Objectives
- Related Work
  - AOP Languages supporting planning
  - EOP Languages
- PLACE
  - Syntax
  - Environment Modeling
  - Planning Mechanism
  - Implementation And GUI
- Case Study
- Conclusion and Perspectives

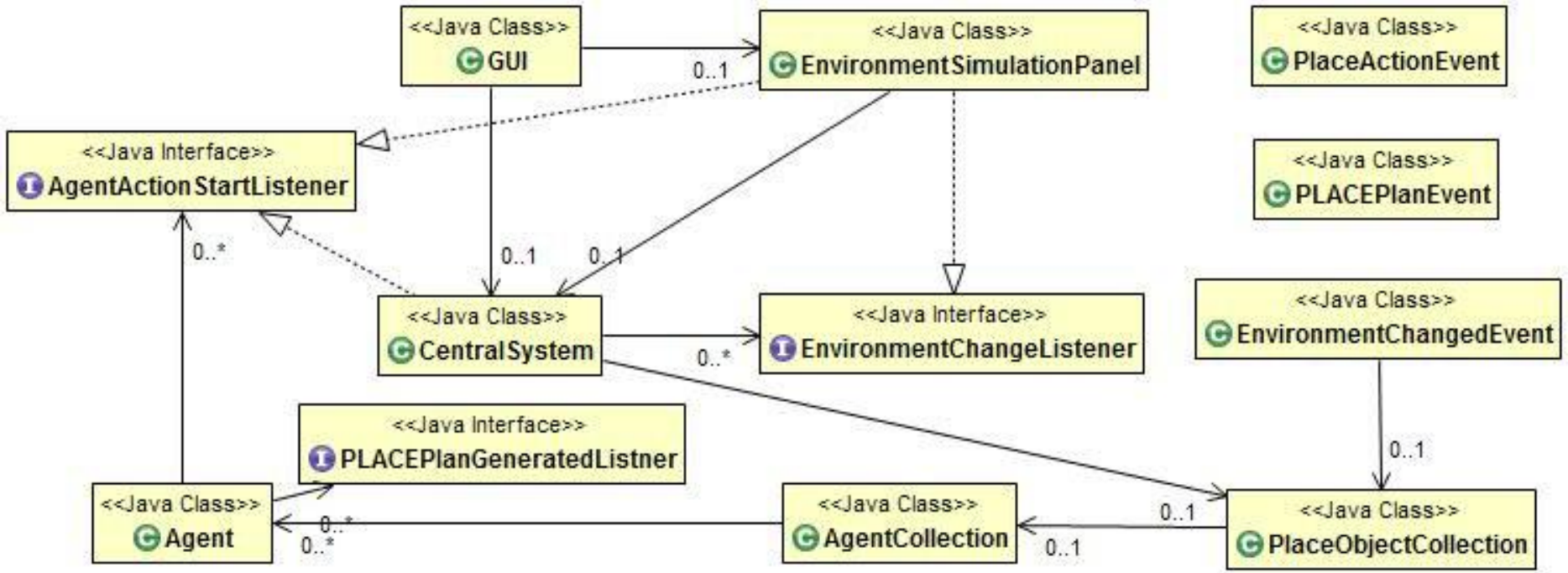
PLACE Working Sequence



PLACE GUI


PLACE
File Run Tools
Compile
temporal
temporal.ep
temporal1.ep
Code Editing
Working Environment
temporal.ep
defineAgent Usman{ /* Comments */
agent_in = {r2}
knowledge=
{
tv_placed(r2);
tv_status(off);
dirty(r1);
dirty(r2);
}
goals={watch(r2);clean(r2);clean(r1);}
actions = {
OP_watchTV{
message = OP_watchTV(?c);
condition = and(hasKnowledge(agent_in(r2)),hasKnowledge(tv_status(on)));
addEffects = {watching(tv)}
deleteEffects = null;
duration=2;
}
OP_TurnOnTv{
message = OP_TurnOnTv(?c);
condition = and(hasKnowledge(agent_in(r2)),hasKnowledge(tv_status(off)));
addEffects = {tv_status(on)}
deleteEffects = {tv_status(off)}
duration=2;
}
OP_TurnOffTv{
message = OP_TurnOffTv(?c);
condition = and(hasKnowledge(agent_in(r2)),hasKnowledge(tv_status(on)),hasKnowledge(watching(tv)));
addEffects = {tv_status(off)}
deleteEffects = {tv_status(on)}
duration=2;
}
OP_cleanRoom{
message = OP_cleanRoom(?c);
condition = and(hasKnowledge(agent_in(?c)),hasKnowledge(dirty(?c)));
Usman
Goals
Knowledge
Proactive Goals
Reactive Goals
Usman's Running Actions
Usman's Initial Temporal Plan
(op_turnontv r2)
(op_cleanroom r2)
(op_watchtv r2)
(op_turnofftv r2)
(op_move r2 r1)
(op_cleanroom r1)
s:0.0 e:2.0 (op_turnontv r2) 2.0
s:2.0 e:4.0 (op_watchtv r2) 2.0
s:4.0 e:6.0 (op_turnofftv r2) 2.0
s:0.0 e:2.0 (op_cleanroom r2) 2.0
s:6.0 e:7.0 (op_move r2 r1) 1.0
s:7.0 e:9.0 (op_cleanroom r1) 2.0
Add Goal:
Proactive
Reactive
Add
Usman: Knowledge
tv_placed(r2)
dirty(r1)
dirty(r2)
clean(?c)
watching(tv)
tv_status(off)
agent_in(r1)
connected(r1,r2)
connected(r2,r1)
Add Knowledge
Status:
clean(?c)
Remove Knowledge
PLACE (Planning based Language for Agents and Computational Environments)
EN
9:35 PM
11/6/2016

PLACE Event System

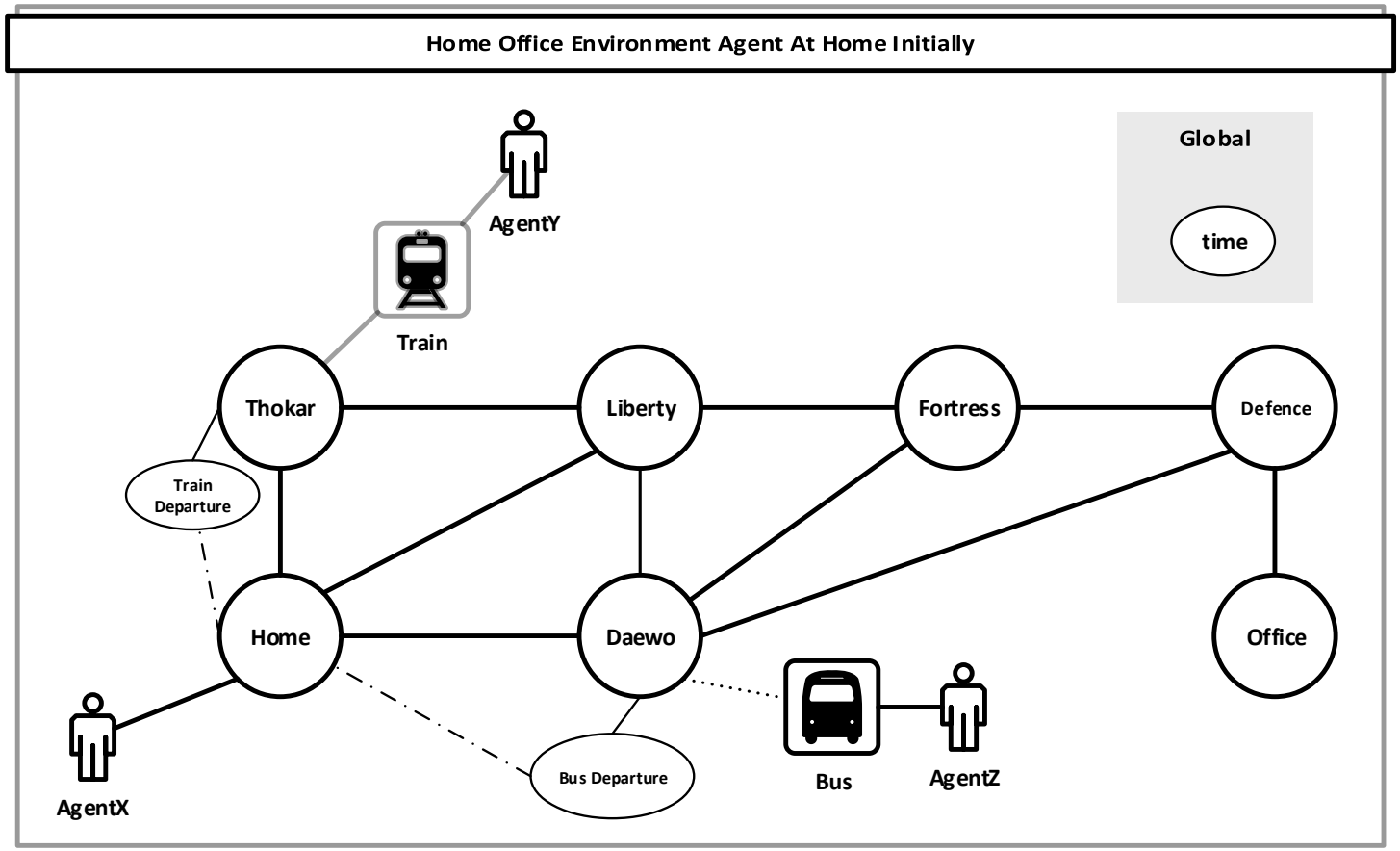


Outline

- Context and Objectives
- Related Work
  - AOP Languages supporting planning
  - EOP Languages
- PLACE
  - Syntax
  - Environment Modeling
  - Planning Mechanism
  - Implementation And GUI
- Case Study
- Conclusion and Perspectives

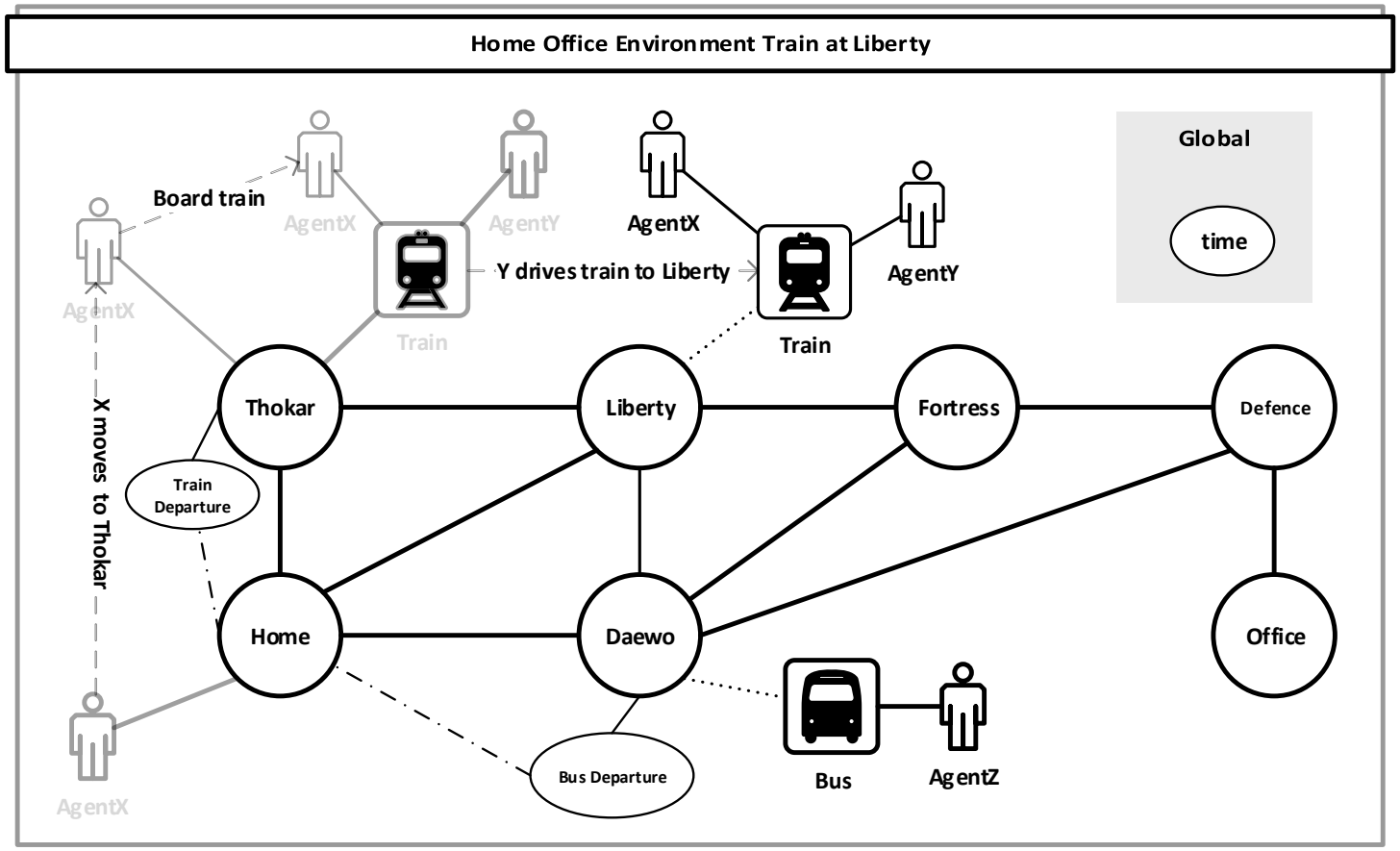
Case Study

Goal: agentIn(Office)



Case Study

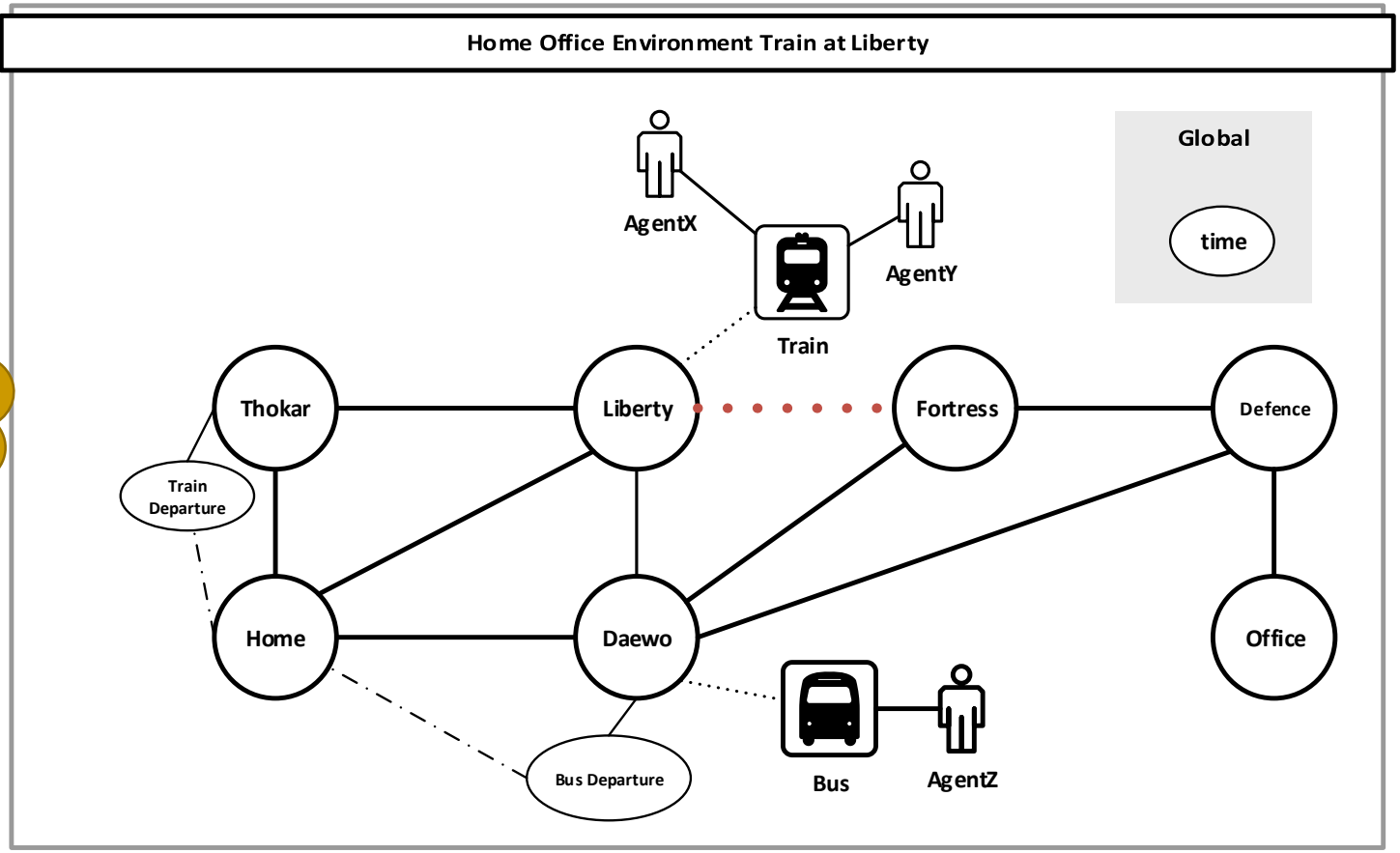
Initial Plan
(Train From Thokar to Defence)



Case Study

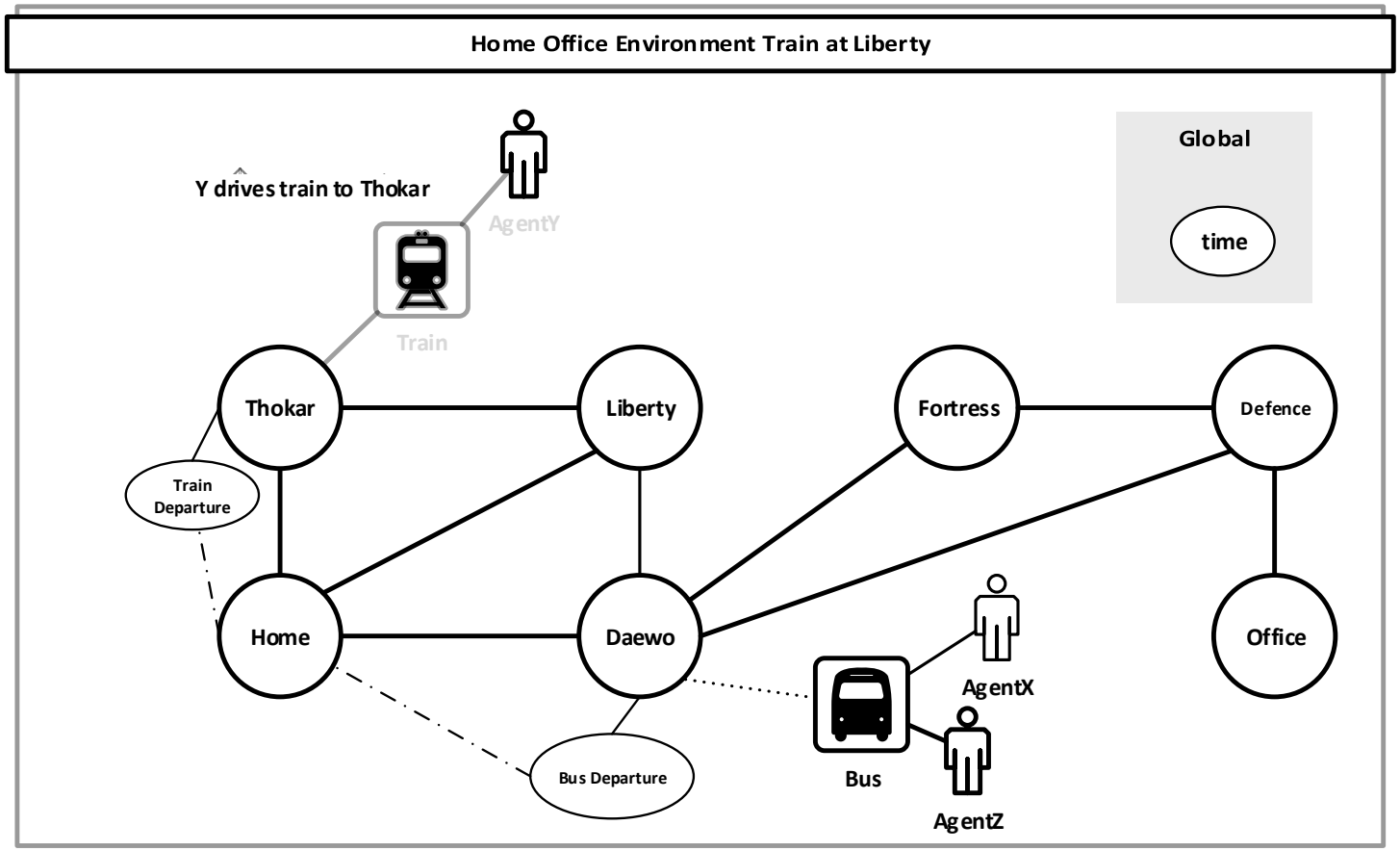
Liberty Fortress Link Breaks Down.

Needs Plan Repairing



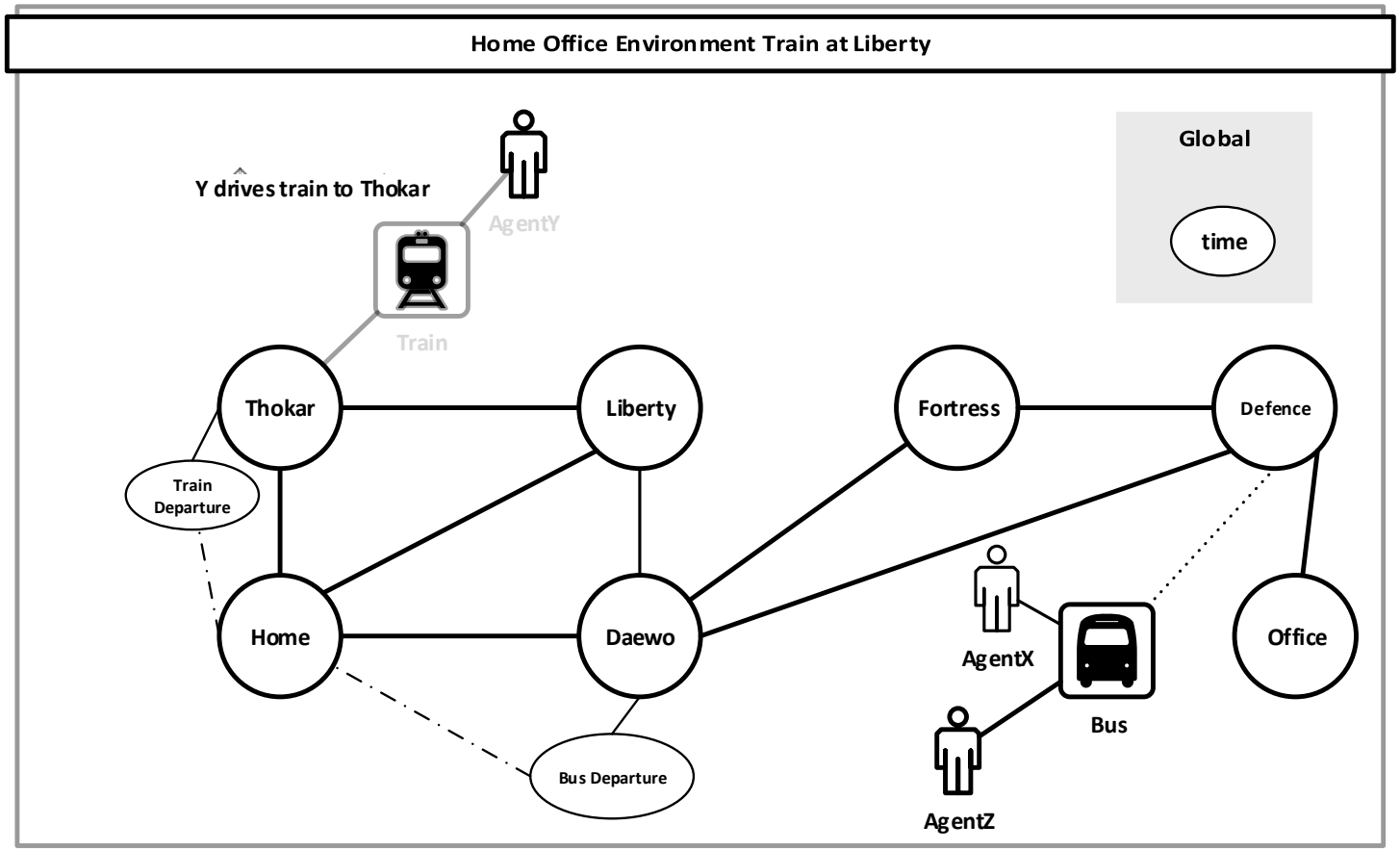
Case Study

Agent Board Bus From Daewo



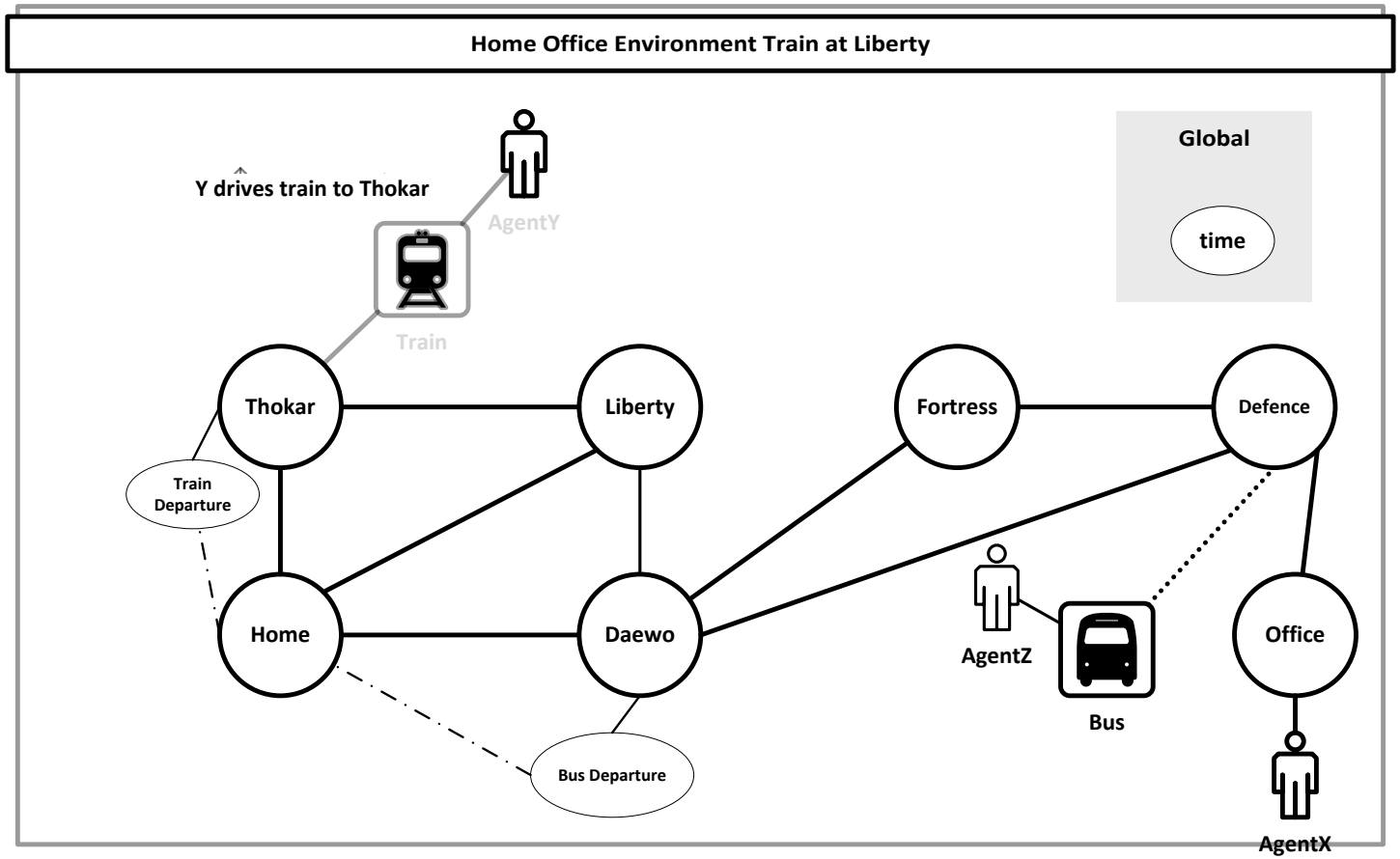
Case Study

AgentZ moves bus from Daewo to Defence



Case Study

agentIn(Office)



Outline

- Context and Objectives
- Related Work
  - AOP Languages supporting planning
  - EOP Languages
- PLACE
  - Syntax
  - Environment Modeling
  - Planning Mechanism
  - Implementation And GUI
- Case Study
- Conclusion and Perspectives

Conclusion & Future Work

- PLACE
  - Planning based Language for Agents and Computational Environments
    - A coherent architecture supporting plan generation, execution and repairing
    - Environment Modeling in AOP
- Future Work
  - Support of goals with deadlines
  - A test-bed for agents simulations

Thanks for your attention!!!