Programming Agent Mobility

Rebecca Montanari, ***Gianluca Tonti***

D.E.I.S. Department
University of Bologna – Italy
{rmontanari,**gtonti**@deis.unibo.it}

Cesare Stefanelli

Department of Engineering
University of Ferrara – Italy
{cstefanelli@ing.unife.it}

Outline

- Motivations
- A policy-based Approach to Mobility
- A policy-controlled Mobile Agent Framework
- Conclusions and Open Issues

Motivations

Mobility adds complexity to the design and development of applications

- Programmers have to explicitly deal with the allocation of agents
- **Mobility specifications embedded into the agent code** are not adequate to cope with the frequent changes of the environment state

⇒ need of flexibility

Question… **HOW to allow dynamic changes in the mobility behaviour of agents without reengineering the agent code?**

…Solution ☑ *Separation of concerns: mobility and application logic*

Policy-based Approach to Mobility

"Policies are rules governing choices in the behavior of a system, and are separated from the components in charge of their interpretation"

IDEA ⇒ **Policies for governing changes in the mobility behaviour of Mobile Agents**

Mobility policies specify:
when, where and which unit of mobility must migrate

Advantages of policy-controlled mobility:

- ☑ Possibility to facilitate dynamic reconfiguration of mobility strategies
- ☑ Possibility to specify mobility behaviour at a higher level of abstraction

Design Requirements

- **Policy language**
  - **declarative**
  - **expressive**
  - **platform-independent**
- **Facilities for policy lifecycle management**
  - ***Specification Service***
  - ***Distribution Service***
  - ***Enforcement Service***
  - **…..**

Mobility Policies

- **Policy language**

☞ **Event-triggered declarative policies for controlled mobility**

⇓

Obligation Ponder Policies

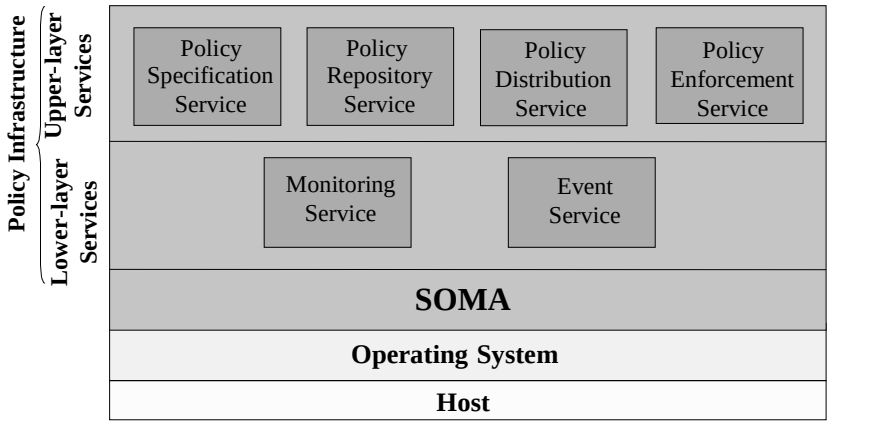
Examples

```
inst oblig  MobPol1 {
  on CPU(load, 90);
  subject s = agents/Buyer;
 do s.go_next_Place(G1, “run”);
 when
MonitoringSystem.isReachable(G1);
}
```

```
inst oblig  MobPol2 {
on CPU(load, 90);
  subject s = System/Relocator;
  target t = agents/Buyer;
  do s.go_next_Place(G1, “run”);
  when
  MonitoringSystem.isReachable(G1);
}
```

Policy-Based Architecture

Facilities for policy lifecycle management



Case Study

Traditionally-programmed MA

```
public class Configurer1 extend Agent {
  Environment env;    VisitedNodes nodes;
  Itinerary itineraryPath;   Place home;
      ….
  void run(){
    if (check_CPU_overloaded) go_next_Place(…);
    check_connection();
    check_local_config();
    if (check_CPU_overloaded) …..
    ToDo_list_elaboration();
    if (check_CPU_overloaded) ….
    SW_configuration();
    upadating_local_vars();
    go_next_Place(…); }
    ….
  void check_connection(){
     Place newConn = env.getConnection();
     if (newConn != null)
          itineraryPath.addLast(newConn); }
        ….
```

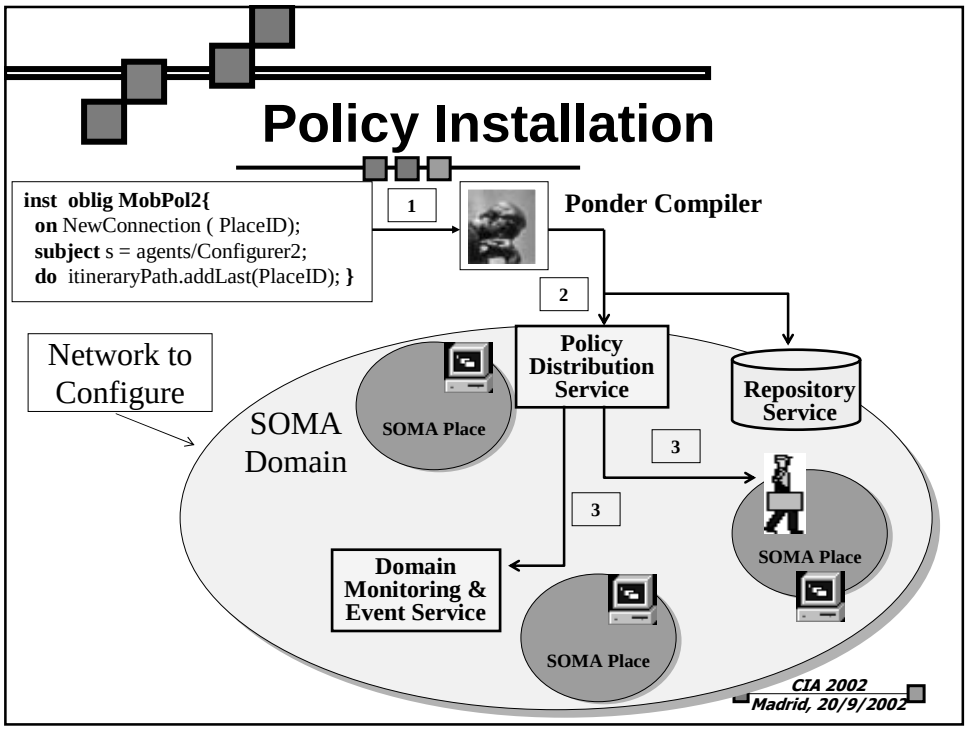
Policy-controlled MA

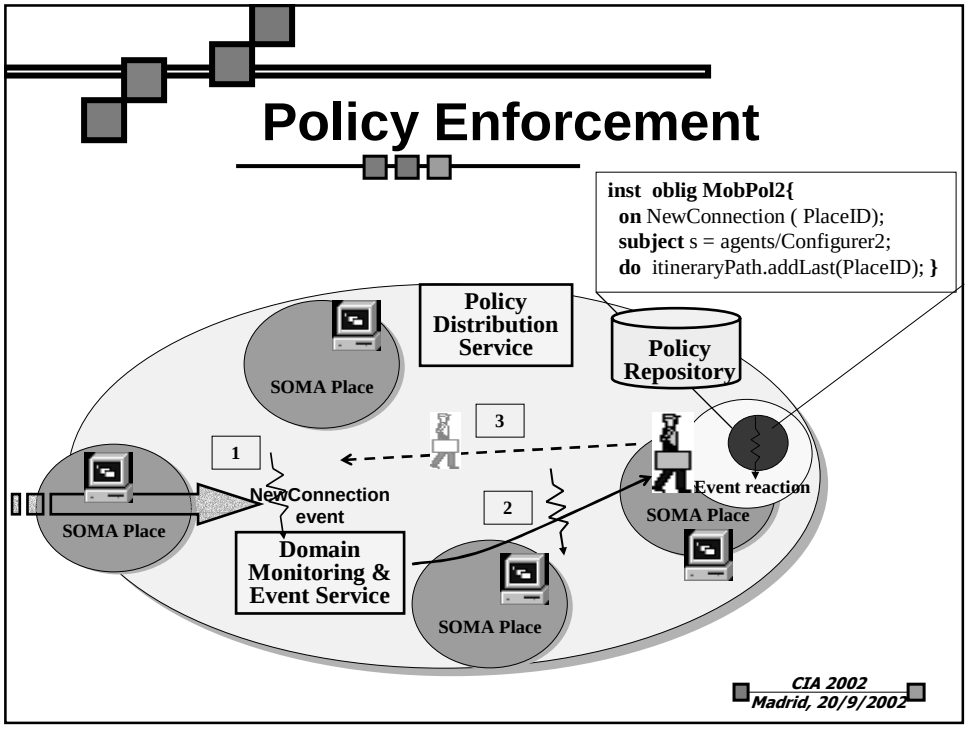
```
class Configurer2 extend PolicyControlledAgent{
  Environment env;    VisitedNodes nodes;
  Itinerary itineraryPath;   Place home;
      ….
  void run(){
    check_local_config();
    ToDo_list_elaboration();
    SW_configuration(); }
    ….
```

Mobility policies

```
inst  oblig MobPol1{
  on TaskCompletion ( s , "run");
  subject s = agents/Configurer2;
  do  go_next_Place(…);
  when  NextNode.isReacheble(); }
```

```
inst  oblig MobPol2{
  on NewConnection ( PlaceID);
  subject s = agents/Configurer2;
  do  itineraryPath.addLast(PlaceID); }
```


Policy Installation
inst oblig MobPol2{
on NewConnection (PlaceID);
subject s = agents/Configurer2;
do itineraryPath.addLast(PlaceID); }
1
Ponder Compiler
2
Network to Configure
SOMA Domain
Policy Distribution Service
Repository Service
SOMA Place
3
3
SOMA Place
Domain Monitoring & Event Service
SOMA Place
CIA 2002
Madrid, 20/9/2002


Policy Enforcement
inst oblig MobPol2{
on NewConnection (PlaceID);
subject s = agents/Configurer2;
do itineraryPath.addLast(PlaceID); }
Policy Distribution Service
Policy Repository
SOMA Place
3
1
NewConnection event
Event reaction
SOMA Place
2
SOMA Place
Domain Monitoring & Event Service
SOMA Place
CIA 2002
Madrid, 20/9/2002

Conclusions

- Policy-based approach to control mobility can simplify mobile application design and maintenance
- Dynamic adaptation of agent mobility behaviour via changes in policy specifications

Open Issues

- Policy-based approach to rule and control other aspects of mobile applications (security, fault tolerance, communication…)
- Policy-based approach to rule and control mobility behaviour of complex systems

Programming Agent Mobility

Rebecca Montanari, ***Gianluca Tonti***

D.E.I.S. Department
University of Bologna – Italy
{rmontanari,**gtonti**@deis.unibo.it}

Cesare Stefanelli

Department of Engineering
University of Ferrara – Italy
{cstefanelli@ing.unife.it}

CIA 2002
Madrid, 20/9/2002