

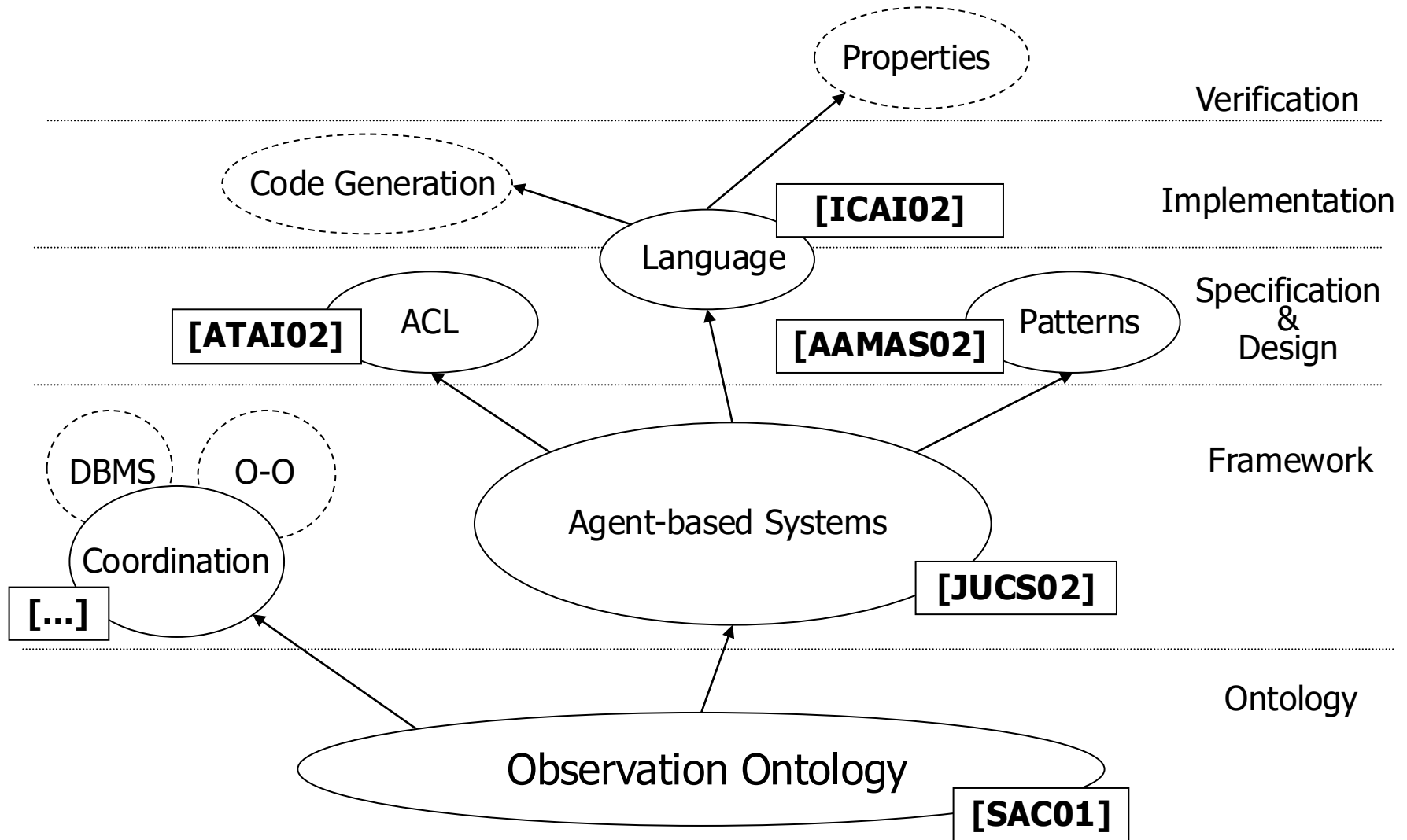
A Specification Language for Agents Observable Behaviour

DEIS - Università degli Studi di Bologna, Cesena, Italy

{mviroli,aomicini}@deis.unibo.it



The Research on Observation



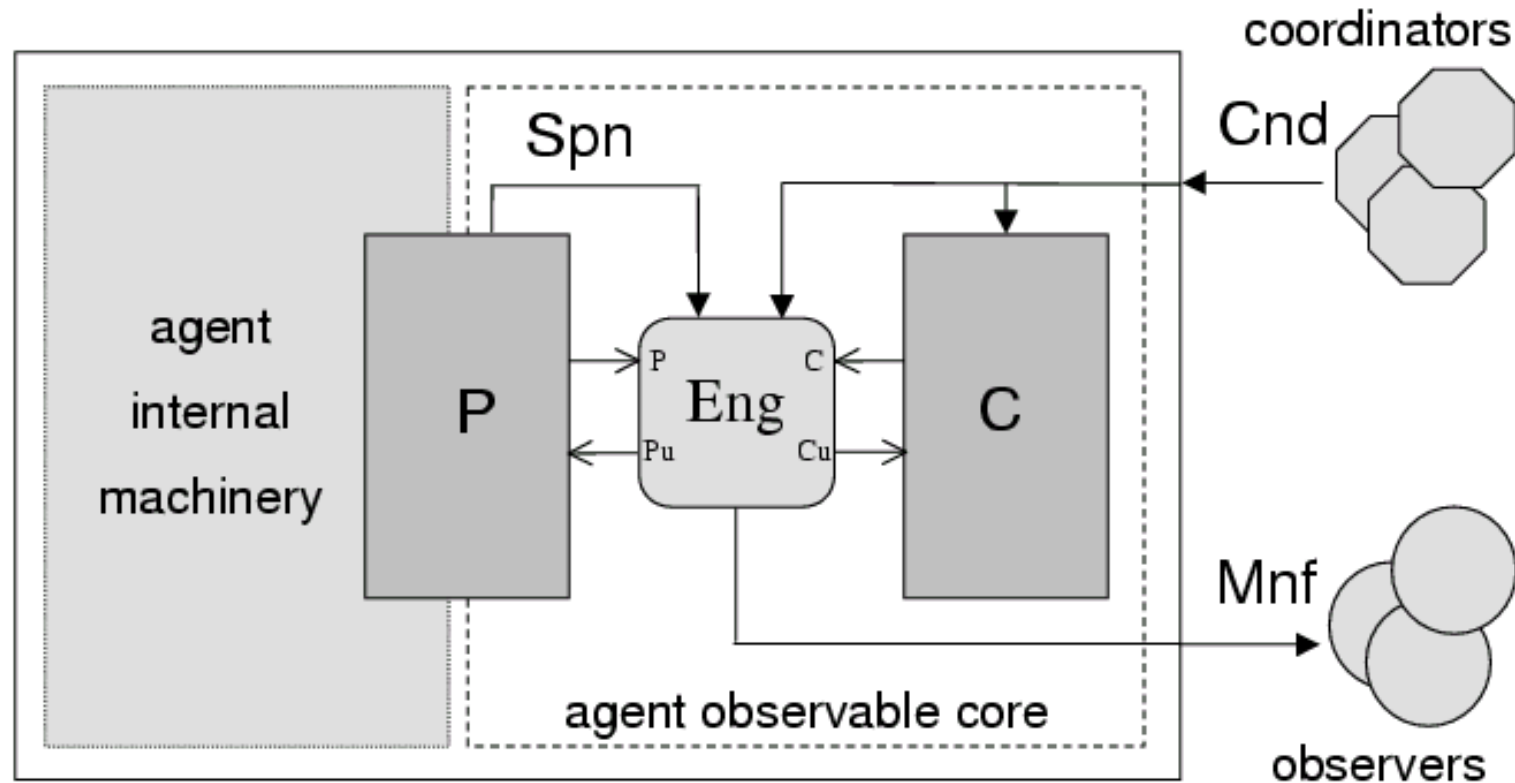
Specification Levels

- White-Box Approach:
 - Complete model of the agent behaviour
 - Suitable for representing an agent internal architecture
- Black-Box Approach:
 - Modelling only agent admissible interaction histories
 - Suitable for representing agent interactions
- Grey-Box Approach:
 - Modelling the agent part responsible for interactions
 - Suitable for representing relationship between agent interactions and agent internal behaviour
 - Abstracting from complicated internal aspects (intelligence,...)

Agents as grey-box abstractions

- Agent model split into:
 - Agent Observable Core: represented
 - Agent Internal Machinery: represented as far as affects the observable core
- Agent Observable Core:
 - Only represents the agent part responsible for the observable behaviour
 - Is a model of the “agent observable behaviour”
- Which general grey-box architecture?
 - we exploit an ontology for the “observation issue”
 - how software components behave when observed?

Agents as Observable Sources



Observation Ontology

- P (place): subpart of the agent status
 - affecting the observable behaviour
 - affected by the internal machinery
- C (configuration):
 - keeps track of the agent interactions
- CND (conditionings): received communicative acts
- SPN (spontaneous moves): internal events
- MNF (manifestations): produced communicative acts
- Eng (core engine):
 - from CND to MNF: reactive behaviour
 - from SPN to MNF: proactive behaviour

Syntax

```
spec ::= agent  $\tau_N$  { import constraints signature structure dynamics }  
import ::= import predicates {  $\sigma_1/i_1, \dots, \sigma_n/i_n$  } from libname  
constraints ::= global conditions {  $\psi_1, \dots, \psi_n$  }  
signature ::= signature {  
    conditionings {  $\kappa_{CND}^1; \dots; \kappa_{CND}^n$  }  
    spontaneous moves {  $\kappa_{SPN}^1; \dots; \kappa_{SPN}^n$  }  
    manifestations {  $\kappa_{MNF}^1; \dots; \kappa_{MNF}^n$  }  
}  
structure ::= structure {  
    place { state {  $\kappa_P$  } initially [  $\tau_P^{in}$  ] updates { transition* } }  
    configuration { state {  $\kappa_C$  } initially [  $\tau_C^{in}$  ] updates { transition* } }  
}  
dynamics ::= dynamics { reactiveness { firing_rule* } proactiveness { firing_rule* } }  
transition ::=  $\tau_{lab}$  : FROM [  $\tau_f$  ] TO [  $\tau_t$  ] conditions;  
firing_rule ::=  $\tau_{trig}$  in [  $\tau_P^0$  ], [  $\tau_C^0$  ] : fires < PU [  $\tau_{PU}$  ], CU [  $\tau_{CU}$  ], MNF [  $\tau_{MNF}$  ] > conditions;  
conditions ::= [ if  $\psi_w^1, \dots, \psi_w^n$  ]  
     $\kappa$  ::=  $\tau$  conditions
```

Semantics

- An agent specification can be compiled into a transition system

$$\langle P \times C, \longrightarrow_{Obs}, (Cnd \times Mnf_{\perp}) \cup (Spn \times Mnf_{\perp}) \rangle$$

- Reactive action:

$$\langle p, c \rangle \xrightarrow{cnd \triangleright mnf_{\perp}}_{Obs} \langle p', c' \rangle$$

- Proactive action:

$$\langle p, c \rangle \xrightarrow{spn \diamond mnf_{\perp}}_{Obs} \langle p', c' \rangle$$

Example

```
agent believer {
  import predicates { inF/1, inV/1, inA/1, isfun/1, funupd/4, funval/3, ismset/1, msetadd/3 } from lib.pl
  constraints { inF(F), inA(A), inV(V), isfun(Fn), isfun(Fn1), ismset(Ms), ismset(Ms1) }
  signature{
    conditionings      { ask(F,A); askwhen(F,A); prop(F,V); }
    spontaneous moves { newK(F,V); eval; }
    manifestations     { send(A,V); sendMany(Ms,V); }
  }
  structure{
    place{
      state {p(Fn,P);} initial [p(null,null)] updates {
        newK(F,V): FROM [p(Fn,P)] TO [p(Fn1,P)] if funupd(Fn,F,V,Fn1);
        add(F,V):  FROM [p(Fn,null)] TO [p(Fn,pr(F,V))];
        eval:      FROM [p(Fn,P)] TO [p(Fn,null)];
      } }
    configuration{
      state {c(Fn)} initial [c(null)] updates {
        askwhen(F,A): FROM [c(Fn)] TO [c(Fn1)] if funval(Fn,F,Ms),msetadd(Ms,A,M2),funupd(Fn,F,Ms,Fn1);
        remall(F):    FROM [c(Fn)] TO [c(Fn1)] if funupd(Fn,F,null,Fn1)
      } } }
  dynamics{
    reactivity{
      ask(F,A) in [p(Fn,P)], [c(Fn1)]: fires <PU [nul], CU [null], MNF [send(A,V)]> if funval(Fn,F,V);
      prop(F,V) in [p(Fn,P)], [c(Fn1)]: fires <PU [add(F,V)], CU [nul], MNF [null]>;
    }
    proactiveness{
      newK(F,V) in [p(Fn,P)], [c(Fn1)]: fires <PU [nul], CU [remall(F)], MNF [sendMany(Ms,V)]> if funval(Fn1,F,Ms);
    }
  } }
```

Bibliography

- [SAC01] M.Viroli, G.Moro, A.Omicini, "On Observation as a Coordination Paradigm: An Ontology and a Formal Framework", Symposium on Applied Computing, SAC 2001.
- [JUCS02] M.Viroli, A.Omicini, "Modelling Agents as Observable Sources", Journal of Universal Computer Science, Springer, 2002.
- [AAMAS02] M.Viroli, A.Omicini, "Specifying Agent Observable Behaviour", Autonomous Agents and Multi-Agent Systems, ACM, 2002.
- [ATAI2002] M.Viroli, A.Omicini, "Towards an Alternative Semantics for FIPA ACL", From Agent Theory to Agent Implementation, EMCSR, 2002.
- [ICAI2002] M.Viroli, A.Omicini, "A Specification Language for Agents Observable Behaviour", International Conference on Artificial Intelligence, CSREA, 2002.