EMAS 2017

DISTRIBUTION TRANSPARENCY IN ENDOGENOUS ENVIRONMENTS: THE JaCaMo CASE

Xavier Limón, Alejandro Guerra-Hernández
Universidad Veracruzana, Xalapa - México

Alessandro Ricci
DISI, University of Bologna, Cesena - Italy

DISTRIBUTION IN MAS

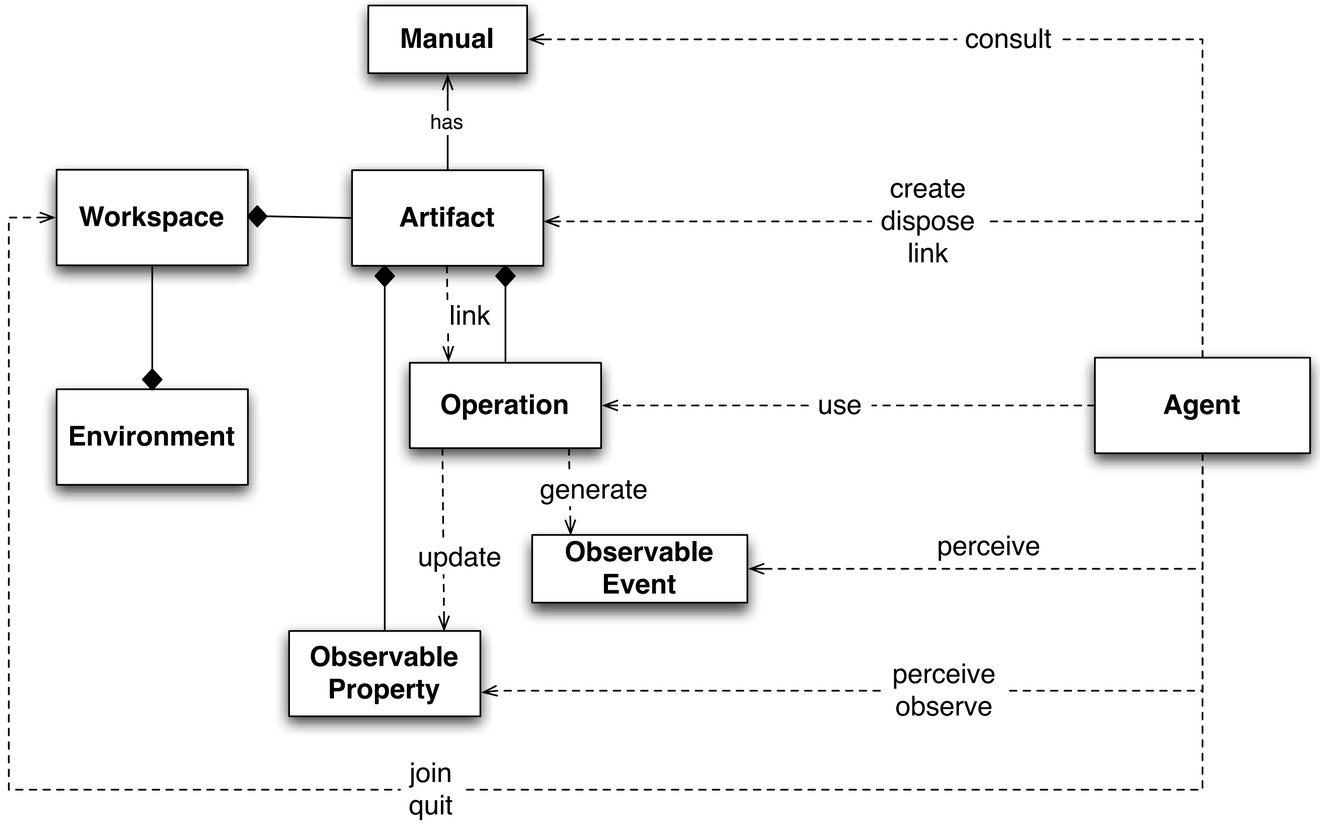
- MAS as Distributed Systems
  - agents running on different network nodes
  - communicating through message passing & ACL
- Assuming a *multi-dimensional view* of MAS
  - **environment distribution**
    - topological abstractions
  - organization distribution…
  - …

ENVIRONMENT AS FIRST-CLASS ABSTRACTION

- Environment for Multi-Agent System community (E4MAS) [Weyns et al]
  - environment as first class abstraction
    - design & architectural level
    - programming level
- In our case
  - A&A conceptual model
  - CArtAgO framework & infrastructure
  - JaCaMo platform

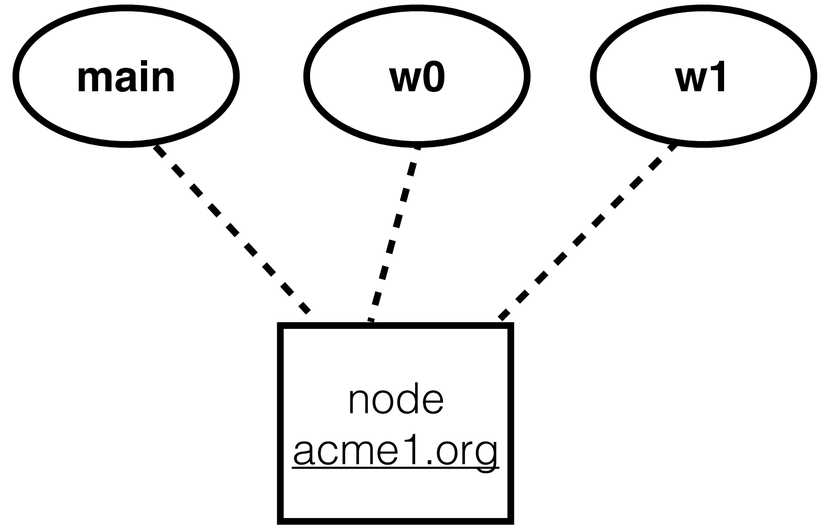
ENVIRONMENT DISTRIBUTION - THE CASE OF A&A -

- Agents & Artifacts (A&A) Conceptual Model
- Distribution model
  - artifacts are stored in workspaces
  - agents can join multiple workspaces
  - workspaces are located in network nodes



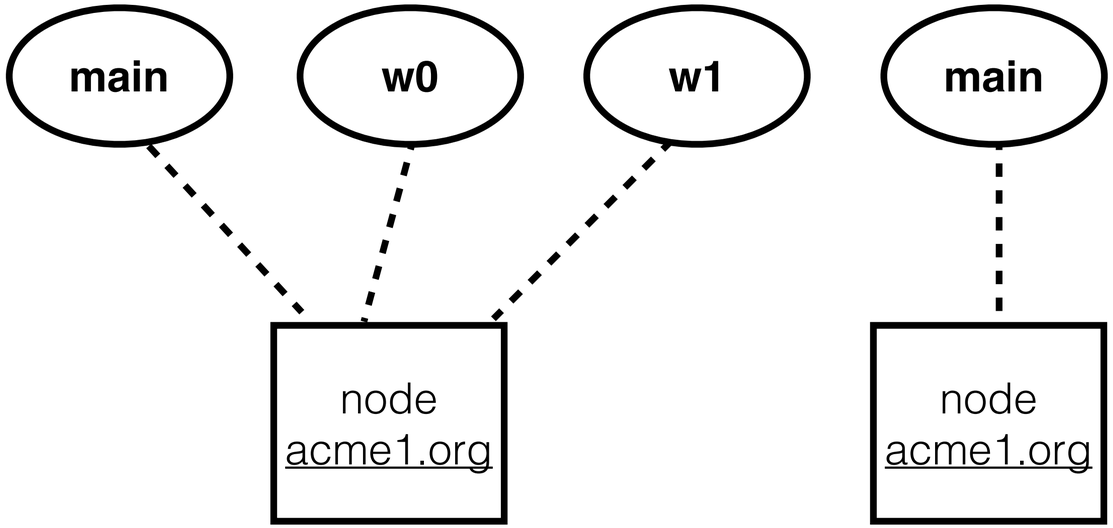
CArtAgO & JaCaMo
- TASTE OF THE API (ABOUT WSP)-

```
!test_wsp.

+!test_wsp
   <- println("creating new workspaces...");
      createWorkspace("w0");
      createWorkspace(“w1”);
      joinWorkspace("w0",WspID0);
      println("hello in ",WspID0);
      makeArtifact("myCount","c4jexamples.Counter",[],ArtId);
      joinWorkspace(w1,WspID1);
      println("hello in ",Name2);
      println("using the artifact of another wsp...");
      inc [artifact_id(ArtId)];
      cartago.set_current_wsp(WspID1);
      println("hello again in ",WspID1);
      println("quit..");
      quitWorkspace;
      ?current_wsp(_,Name3,_);
      println("back in ",Name3).
```



CArtAgO & JaCaMo
- TASTE OF THE API (ABOUT WSP)-

```
!test_remote.

+!test_remote
   <- ?current_wsp(Id,_,_);
      +default_wsp(Id);
      println("testing remote..");
      joinRemoteWorkspace(“main","acme1.org",WspID2);
      ?current_wsp(_,WName,_);
      println("hello there ",WName);
      !use_remote;
      quitWorkspace.

+!use_remote
   <- makeArtifact("c0","examples.Counter",[],Id);
      focus(Id);
      inc;
      inc.
```



(MAIN) LIMITATIONS

- API level (agent programmer level)
  - mixing the logical (workspaces) and physical (IP address) layers in the source code
    - i.e. joinWorkspace and joinRemoteWorkspace
- Modeling/Design & infrastructural level
  - environment as a flat set of workspaces
    - not ideal for modelling complex, structured, possibly large environments

EXTENDING THE MODEL

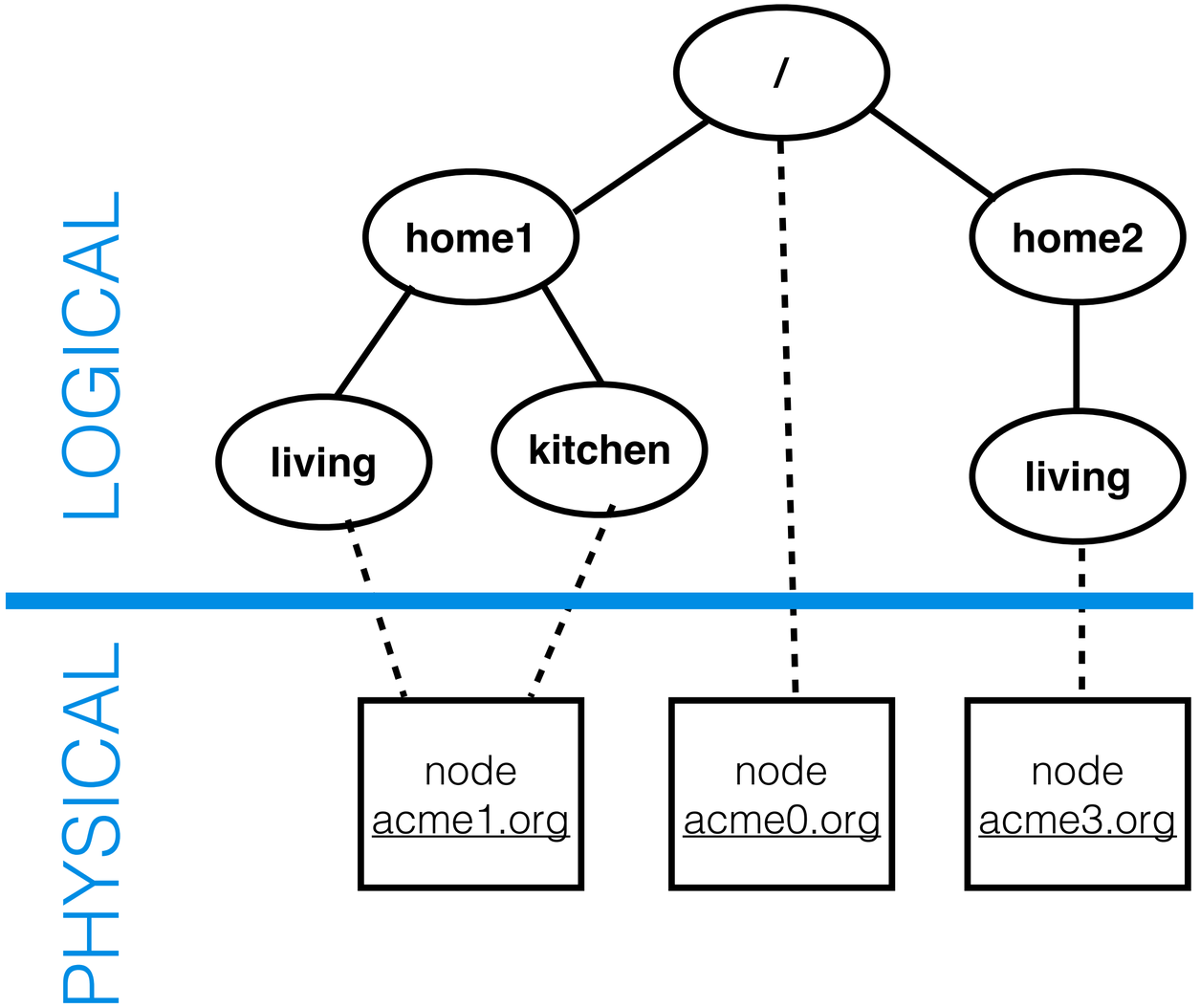
- Uniformity & distribution transparency
- Hierarchical structure

EXTENDING THE MODEL

- **Full distribution transparency**
  - stronger separation between logical level (API) & physical level
    - joinWorkspace only, no more joinRemoteWorkspace
  - nodes appear only when creating/configuring the workspaces
    - as a property specifying the location

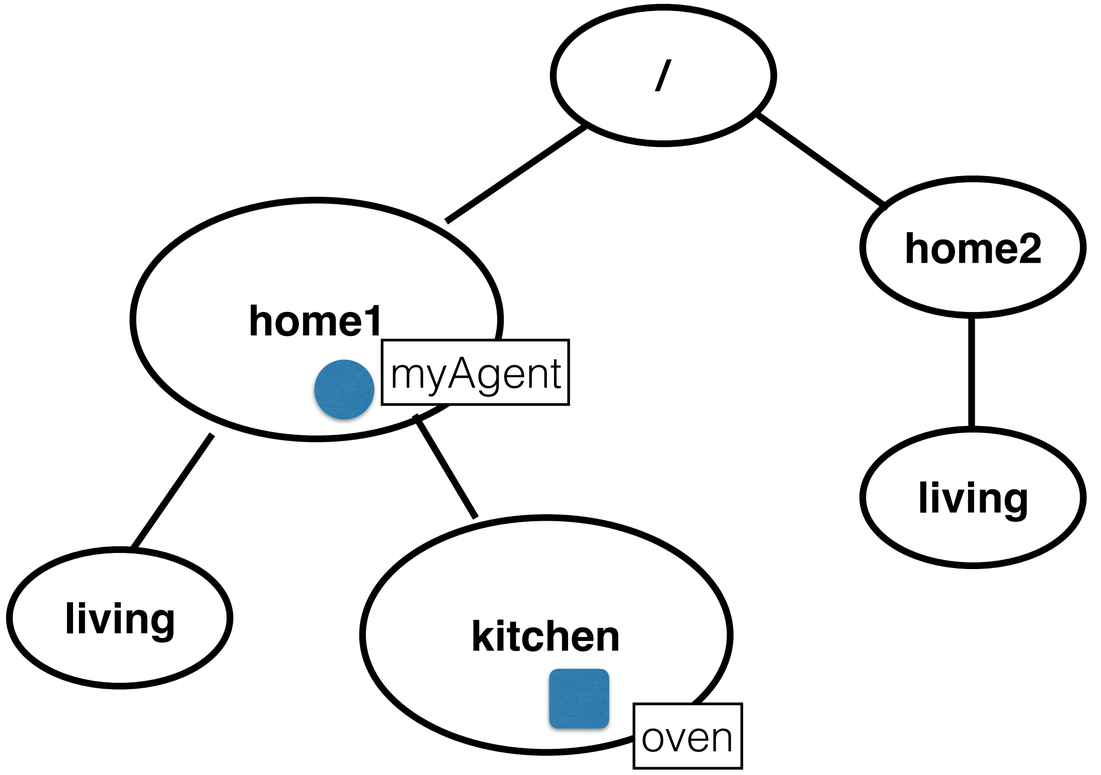
EXTENDING THE MODEL

- **Hierarchical structure**
  - adopting a file system like approach to structure the whose set of workspaces of an environment/MAS
    - / is the root (main) workspace
  - names
    - /home1/living (wsp)
    - /home1/kitchen/oven (artifact)



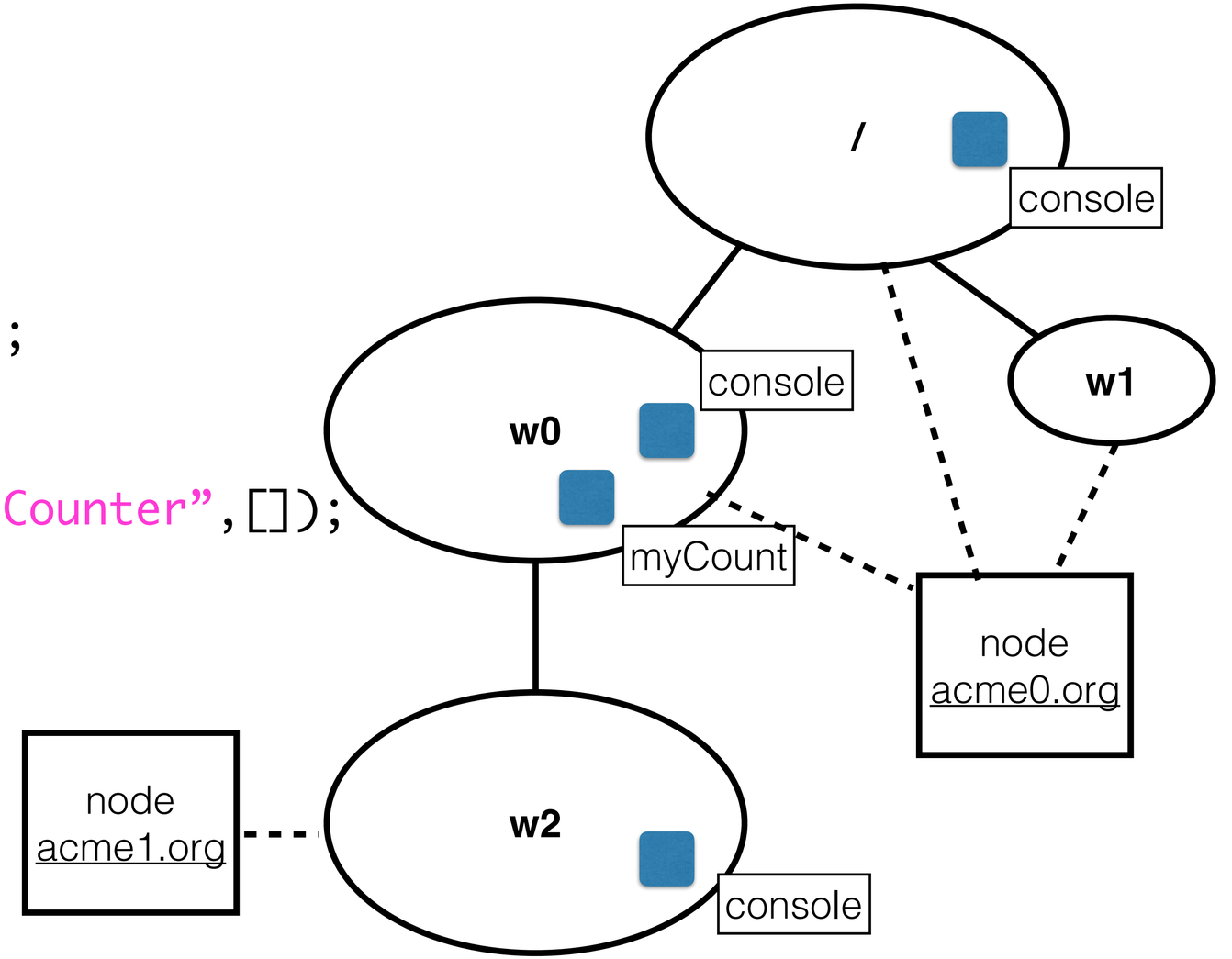
EXTENDING THE MODEL

- **Absolute** and **relative** names
  - *home* workspace = workspace where the agent booted (when entering the MAS)
    - e.g. home1
  - using relative names
    - `joinWorkspace(kitchen)`
    - `focus(kitchen.oven)`



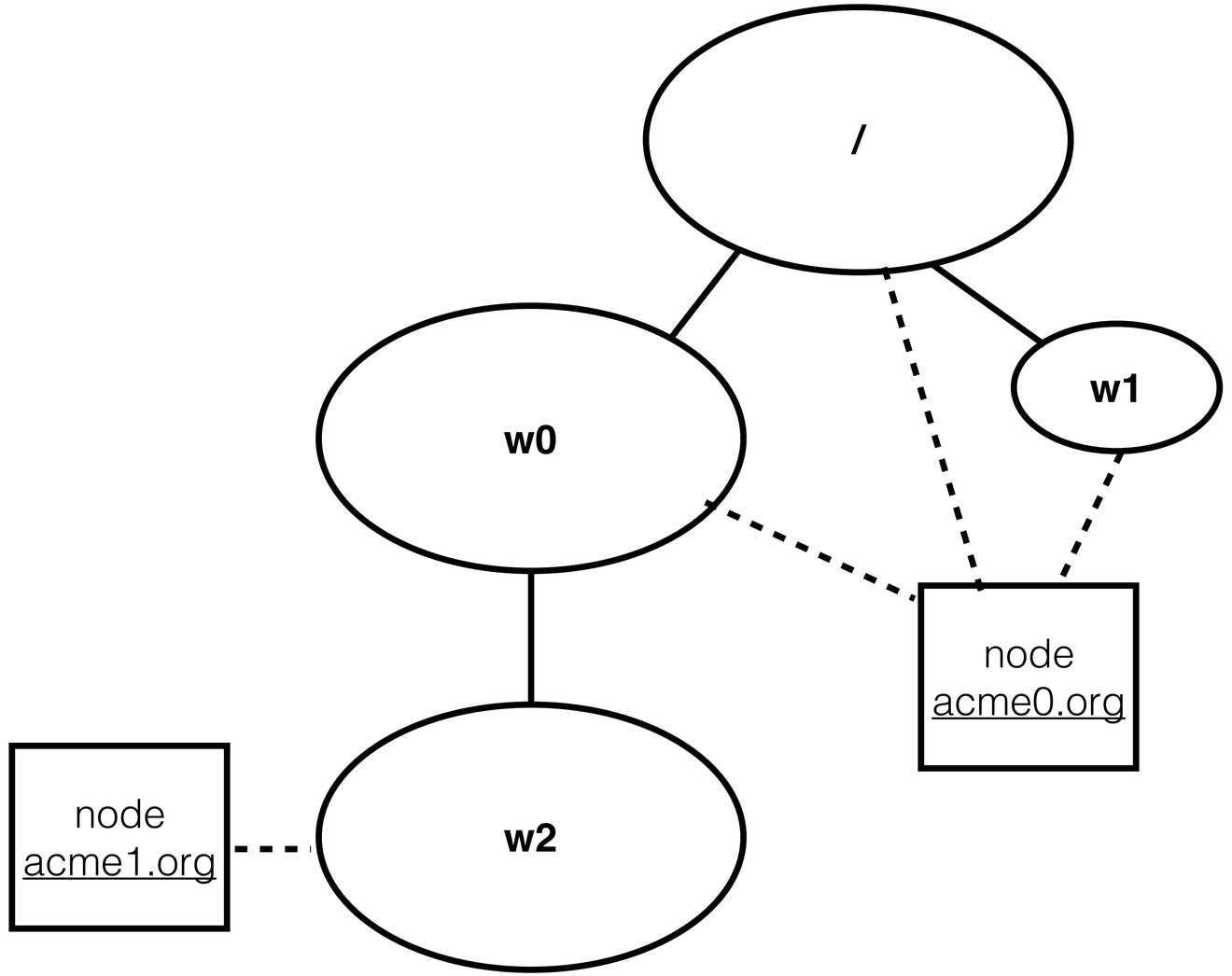
TASTE OF NEW API

```
!test_wsp.

+!test_wsp
   <- println("creating new workspaces...");
      joinWorkspace(w0);
      println("hello in w0")[wsp(w0)];
      makeArtifact(w0.myCount,"c4jexamples.Counter",[]);
      inc [art(w0.myCount)];
      joinWorkspace(w0.w2);
      println("hello in w2")[wsp(w0.w2)];
      println("quit from w2");
      quitWorkspace(w0.w2).
```



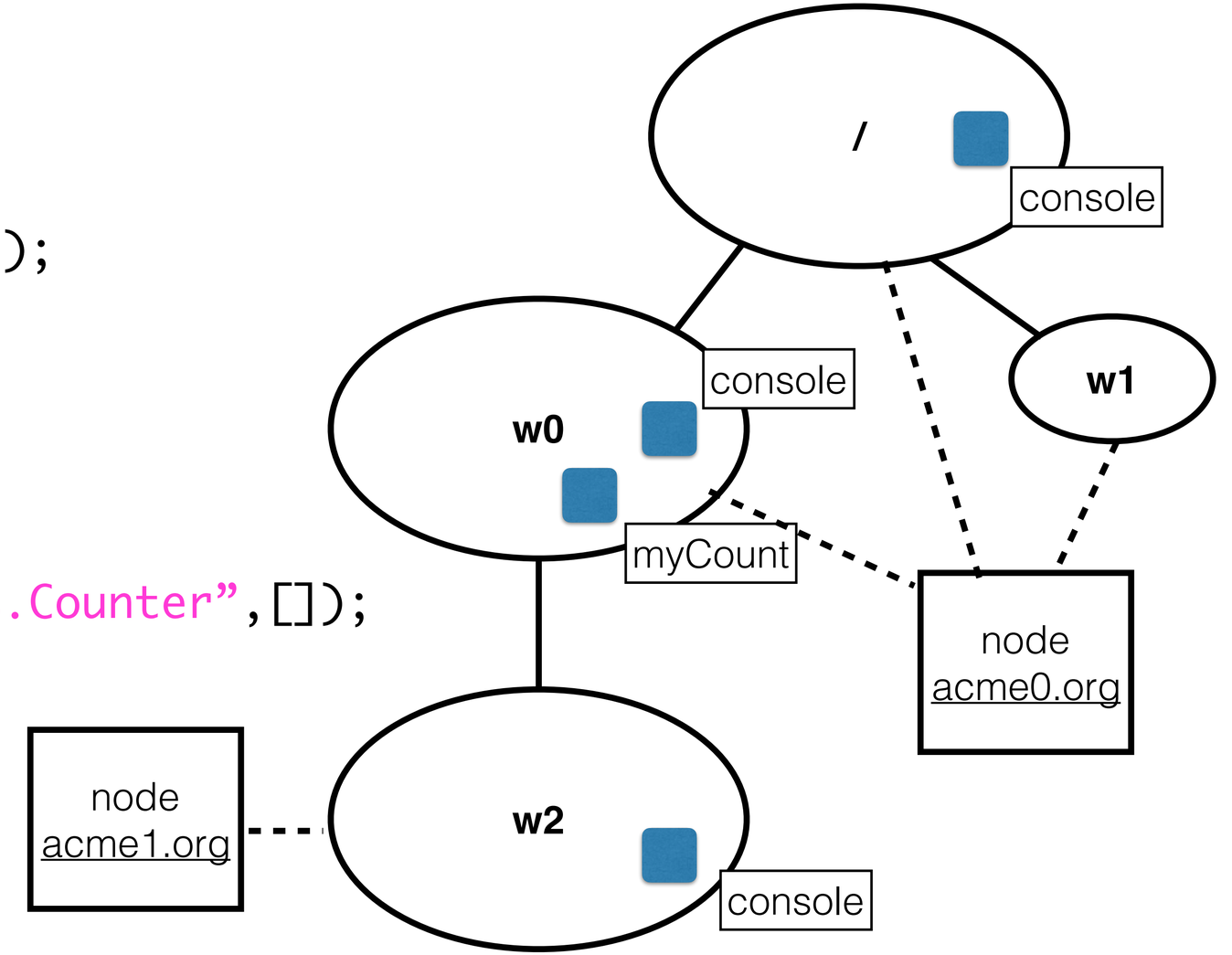
SPECIFYING THE PHYSICAL LOCATION
- JaCaMO MAIN/CONFIG FILE -

```
mas house {

   workspace w0 { … }

   workspace w1 { … }

   workspace w0.w2 {
     located_at n1
   }
}

node n1 acme1.org
```



TASTE OF NEW THE API
- DYNAMIC WSP CREATION -

```
!test_wsp.

+!test_wsp
   <- println("creating new workspaces...");
      createWorkspace(w0);
      createWorkspace(w1);
      createWorkspace(w0.w2, “acme1.org”);
      joinWorkspace(w0);
      println("hello in w0”)[wsp(w0)];
      makeArtifact(w0.myCount,”c4jexamples.Counter”,[]);
      inc [art(w0.myCount)];
      joinWorkspace(w0.w2);
      println("hello in w2”)[wsp(w0.w2)];
      println("quit..");
      quitWorkspace(w0.w2).
```



BENEFITS

- Uniformity
  - single workspace abstraction to model the full topology
- Distribution transparency
  - no need to specify physical addresses when joining a workspace
  - workspaces can be physically moved without impacting on the source code

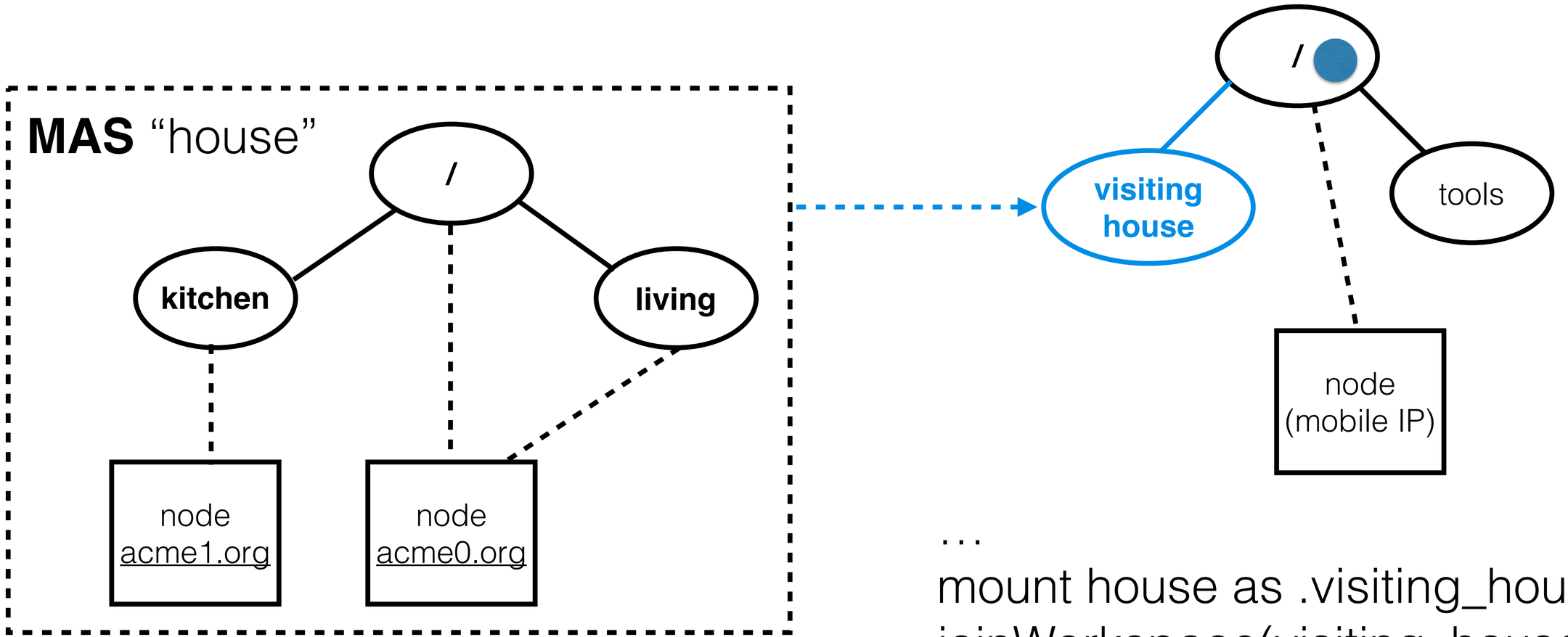
FIRST IMPLEMENTATION

- First prototype implementing the new model
  - branch on the github CArtAgO repository
- Simple centralised management strategy
  - one node is used as the main node of the MAS
    - single point keeping track of the whole MAS configuration (set of workspaces and their location)
  - every joinWorkspace/createWorkspace/quitWorkspace issued by an agent from any other node of the MAS is dispatched to this node
  - but all nodes keep a copy of the topology information
    - dispatched by the central node when it is updated

ONGOING / NEXT STEPS

- Improving & refining the implementation
  - performance evaluation
- **Fault tolerance & resiliency**
  - exploring Erlang/AKKA (actor) - like models
    - making it observable to agents (through artifacts) the state of workspace - if it is reachable or not (crashed/down)
- **Dynamic *mounting***
  - To attach or mount (a MAS) to an existing MAS in execution, at runtime
    - mobile apps

MOUNTING - THE IDEA



…
mount house as .visiting_house
joinWorkspace(visiting_house.kitchen)
…

THANKS.