

Practical Session

Programming Multiagent Systems
WESAAC 2013

Olivier Boissier	(EMSE, France)
Rafael H. Bordini	(PUC-RS, Brazil)
Jomi F. Hübner	(UFSC, Brazil)
Alessandro Ricci	(Bologna University, Italy)

Scenario (I)

- Giacomo wants to build a house
- We consider two main phases:
 1. Contracting specialised companies
Giacomo **hires** various companies specialised in different aspects of construction
 2. Building the house
Contractors **execute** the main workflow for building the house under Giacomo's supervision

Scenario (2)

- Phase I: Contracting specialised companies
- The objective here is to hire one company for each of these tasks:
 - (a) Site preparation
 - (b) Lay floors
 - (c) Build walls
 - (d) Build roof
 - (e) Fit windows
 - (f) Fit doors
 - (g) Install the plumbing
 - (h) Install the electrical system
 - (i) Paint the exterior of the house
 - (j) Paint the interior of the house

NB: The same company can be hired for more than 1 task

Scenario (3)

- Phase 2: Building the house
 - After the companies have been hired, they have to execute their tasks on time and in coordination with each other
 - Some tasks depend on others and some tasks can be done in parallel, as represented by the workflow (";" for sequence and "|" for parallel)

a ; b ; c ; (d | e | f) ; (g | h | i) ; j

Scenario (3)

- Phase 2: Building the house

- After the companies have been hired, they have to execute their tasks in a sequence and in coordination with each other.
- Some tasks depend on other tasks. Some tasks can be done in parallel. The workflow is represented by the workflow diagram below, where "S" for sequence and "|" for parallel.

- (a) Site preparation
- (b) Lay floors
- (c) Build walls
- (d) Build roof
- (e) Fit windows
- (f) Fit doors
- (g) Install the plumbing
- (h) Install the electrical system
- (i) Paint the exterior of the house
- (j) Paint the interior of the house

a ; b ; c ; (d | e | f) ; (g | h | i) ; j

An MAS for House Building

- To make the scenario suitable for a course like this, we introduced some simplifications in the system design
- We summarise the solution using our approach, commenting on:
 - Agents
 - Contracting Phase (using Environment)
 - Building Phase (using Organisation)

Agents

- Two types of agents:
 - **House owner** (Giacomo): provides the requirements for the house, with budget limitations
 - **Companies**: they will offer their service and, if hired, will execute the house building tasks; they are characterised by their competences in house building

Agents Implementation

- The house owner is programmed in Jason
- Companies are programmed in Jason
(or 2APL, Jadex, ...)
- A heterogeneous agent system

Contracting Phase

- Electronic **auctions** will be used to hire the required companies
- One auction for each task
- Each auction is started with:
 - the task description
 - the maximum value the owner can pay for it
- By the end of an auction, the company to be hired for that task is determined

Environment Implementation

- An **auction artifact** encapsulates the auction mechanism
- Giacomo creates instances of such artifacts for creating/managing the various auctions; one such auction is used for hiring companies for each of the house building tasks
- Companies can perceive those artifacts and bid according to their competence and following their own strategies
- After some time Giacomo decides to finish the auction, observing the current best bid shown on the artifact

The Auction Artifact

- observable properties:
 - task description
 - maximum payment value
 - the current best bid (lower service price)
 - the current winning agent ID
- operation:
 - `bid(p)`: places a new bid for doing the service for price `p` (used by company agents to bid in a given auction)

src/java/tools/AuctionArt.java

```
/**
 *      Artifact that implements the auction.
 */
public class AuctionArt extends Artifact {

    @OPERATION public void init(String taskDs, int maxValue) {
        // observable properties
        defineObsProperty("task", taskDs);
        defineObsProperty("maxValue", maxValue);
        defineObsProperty("currentBid", maxValue);
        defineObsProperty("currentWinner", "no_winner");
    }

    @OPERATION public void bid(double bidValue) {
        ObsProperty opCurrentValue = getObsProperty("currentBid");
        ObsProperty opCurrentWinner = getObsProperty("currentWinner");
        if (bidValue < opCurrentValue.intValue()) { // the bid is better
            opCurrentValue.updateValue(bidValue);
            opCurrentWinner.updateValue(getOpUserName());
        }
    }
}
```

Contracting: Agent Side

- Giacomo has plans to launch all auctions by creating the corresponding artifacts
- Company agents have plans to look for the auction artifacts of their interest and plans defining their own bidding strategy
- After some time Giacomo looks at the best bid in each auction artifact and awards a contract for the winning company

src/asl/giacomo.asl

```
/* Initial goal */
```

```
!have_a_house.
```

```
/* Plans */
```

```
+!have_a_house
```

```
    <- !contract; // hire the companies that will build the house  
        !execute. // (simulates) the execution of the construction
```

```
/* Plans for Contracting */
```

```
+!contract
```

```
    <- !create_auction_artifacts;  
        !wait_for_bids.
```


src/asl/giacomo.asl

```
+!create_auction_artifacts
  <- !create_auction_artifact("SitePreparation", 2000); // 2000 is the max
      !create_auction_artifact("Floors", 1000);
      !create_auction_artifact("Walls", 1000);
      !create_auction_artifact("Roof", 2000);
      !create_auction_artifact("WindowsDoors", 2500);
      !create_auction_artifact("Plumbing", 500);
      !create_auction_artifact("ElectricalSystem", 500);
      !create_auction_artifact("Painting", 1200).

+!create_auction_artifact(Task,MaxPrice)
  <- .concat("auction_for_",Task,ArtName);
      makeArtifact(ArtName, "tools.AuctionArt", [Task, MaxPrice], ArtId);
      focus(ArtId).

-!create_auction_artifact(Task,MaxPrice)[error_code(Code)]
  <- .print("Error creating artifact ", Code).

+!wait_for_bids
  <- println("Waiting the bids for 5 seconds...");
      .wait(5000); // use an internal deadline of 5 seconds to close the auc
      !show_winners.
```

src/asl/companyA.asl

```
// This company bids for Plumbing only
// Strategy: fixed price

{ include("common.asl") }

my_price(300). // initial belief

!discover_art("auction_for_Plumbing").

+currentBid(V)[artifact_id(Art)] // there is a new value for current bid
  : not i_am_winning(Art) &      // I am not the current winner
    my_price(P) & P < V           // I can offer a better bid
  <- // .print("my bid in auction artifact ", Art, " is ", P);
     bid( P ).                   // place my bid offering a cheaper service

/* plans for execution phase */

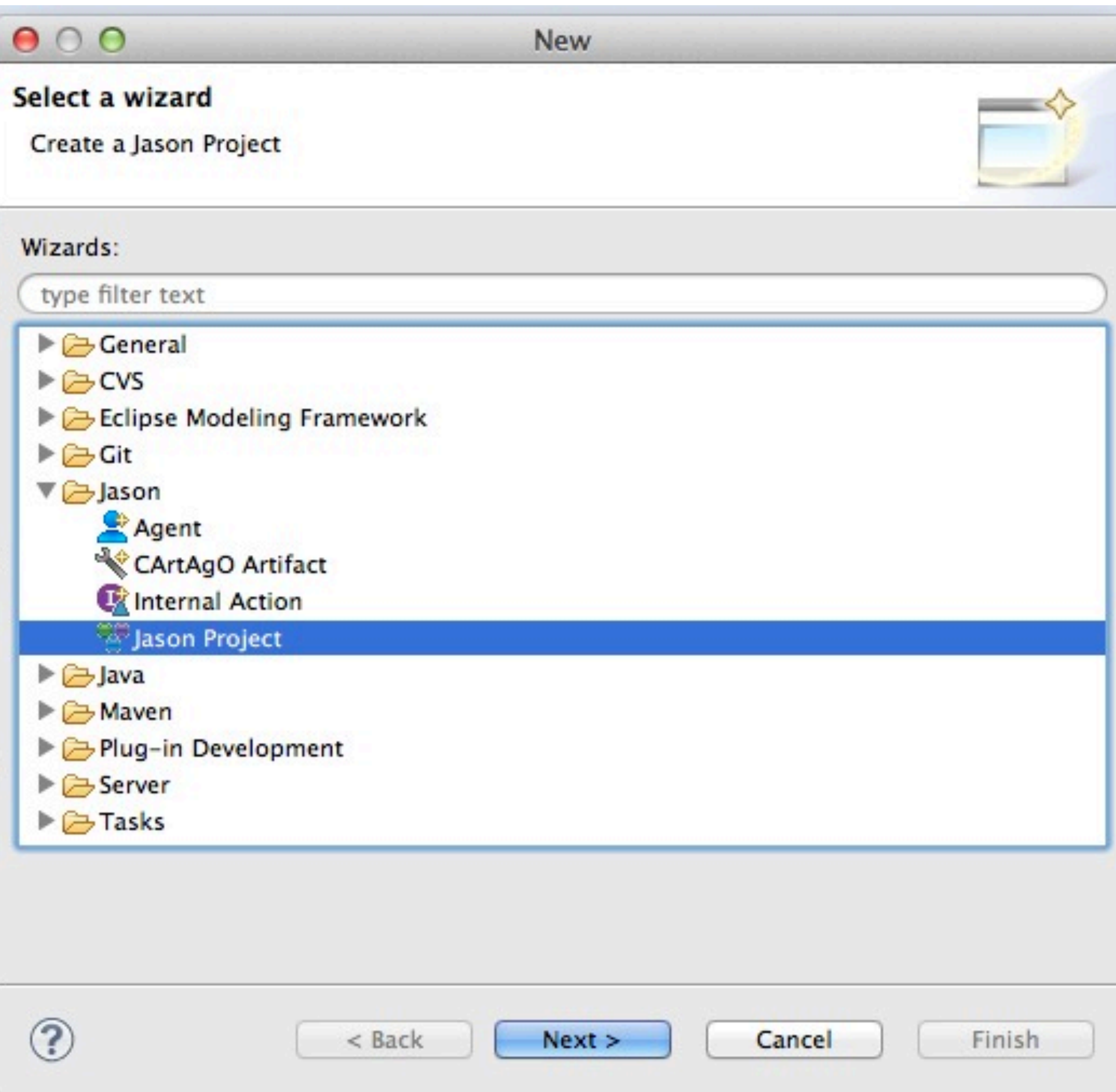
{ include("org_code.asl") }

// plan to execute organisational goals (not implemented)
|
+!plumbing_installed // the organisational goal (created from an obligation)
  <- installPlumbing. // simulates the action (in GUI artifact)
```

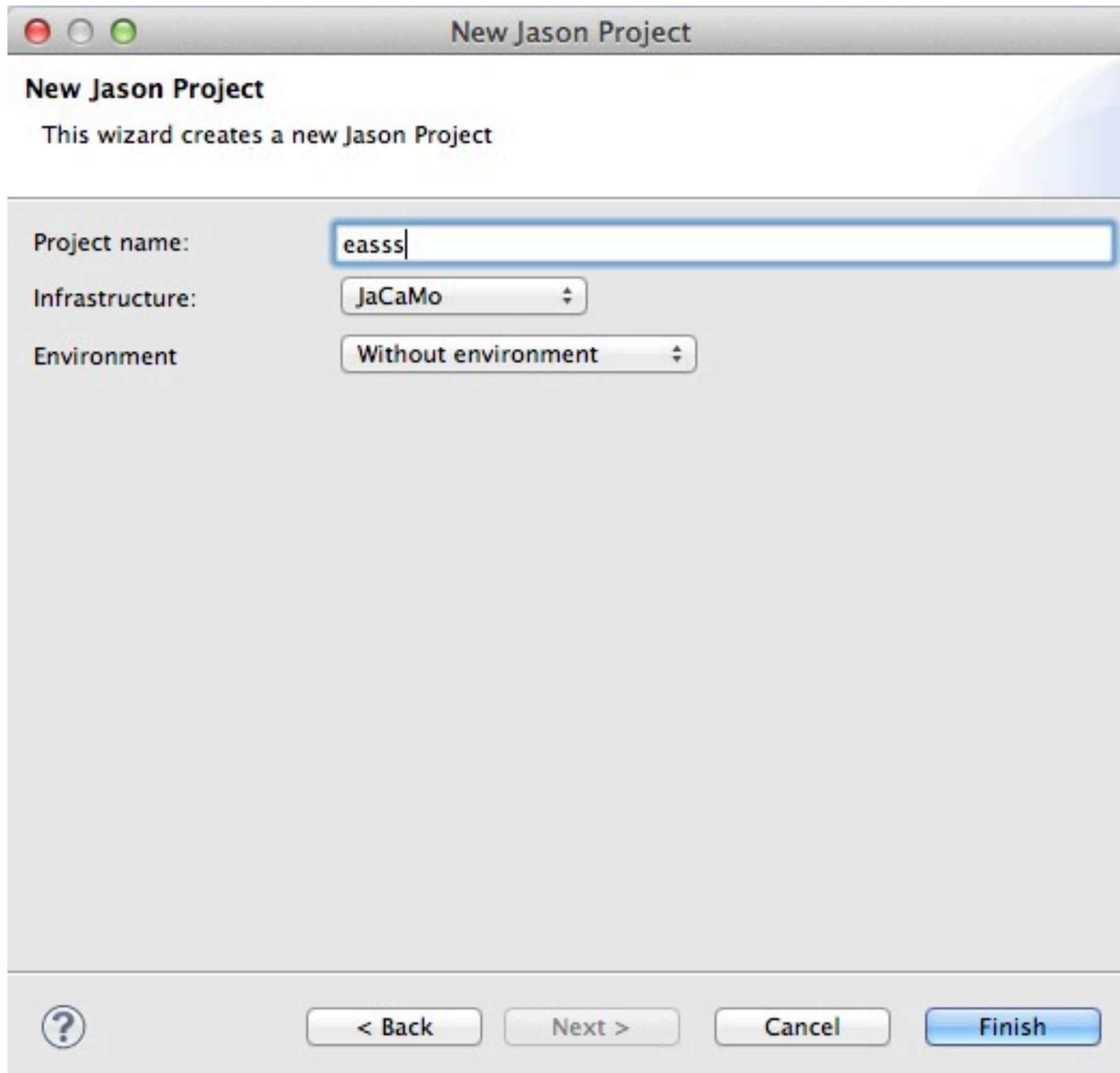

Eclipse Project

Requirements

- Eclipse \geq Indigo
- Jason \geq 1.3.7 ([link](#))
- Jason Eclipse plugin ([link](#))
- Initial code of the application ([link](#))



JaCaMo infrastructure



The image shows a screenshot of a macOS-style window titled "New Jason Project". The window has a title bar with red, yellow, and green window control buttons. Below the title bar, the text "New Jason Project" is displayed, followed by a subtitle: "This wizard creates a new Jason Project". The main area of the window contains three labeled fields: "Project name:" with a text input field containing "easss", "Infrastructure:" with a dropdown menu showing "JaCaMo", and "Environment" with a dropdown menu showing "Without environment". At the bottom of the window, there is a footer bar containing a help icon (a question mark in a circle) on the left, and four buttons: "< Back", "Next >", "Cancel", and "Finish" on the right. The "Finish" button is highlighted with a blue gradient.

New Jason Project

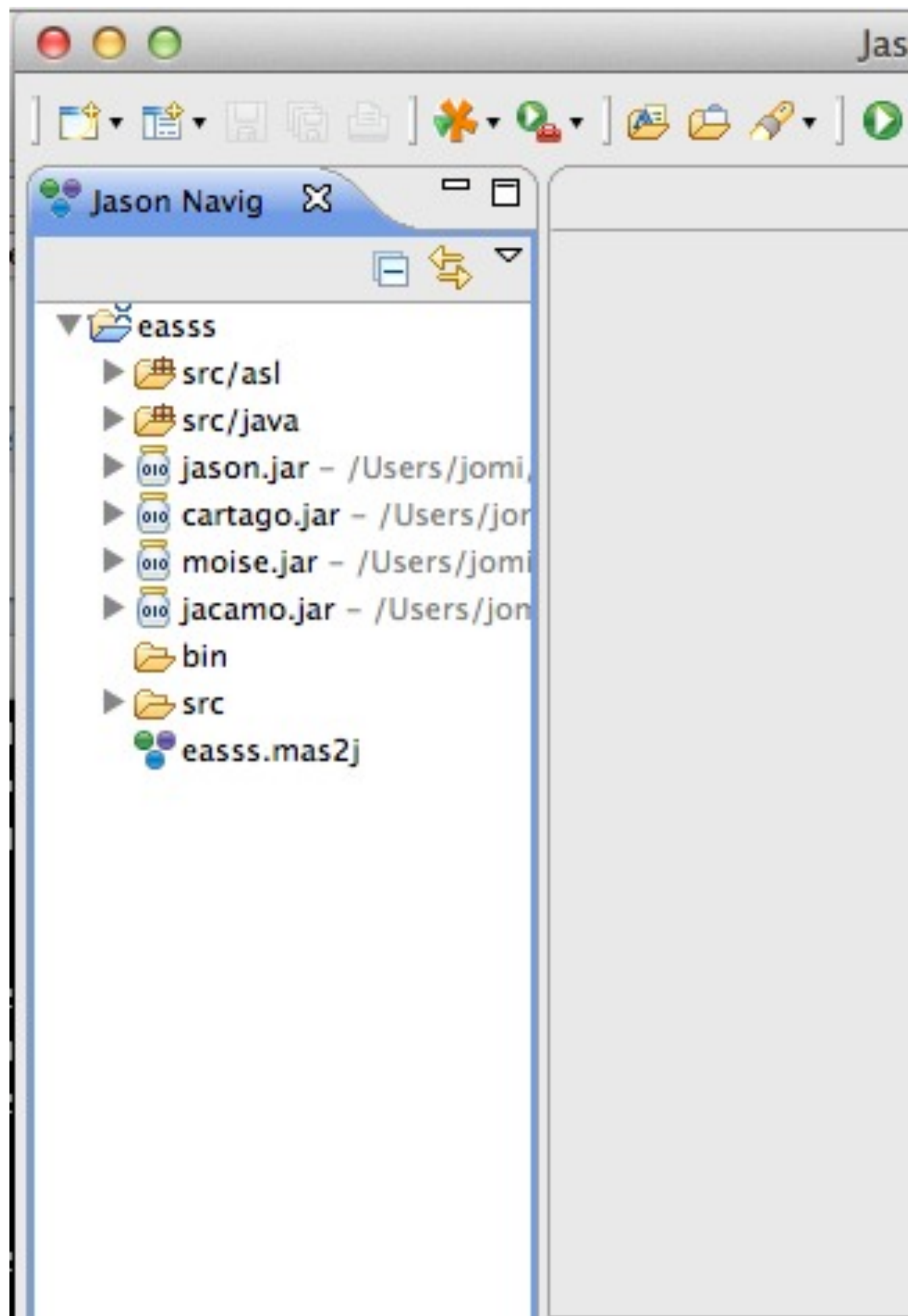
This wizard creates a new Jason Project

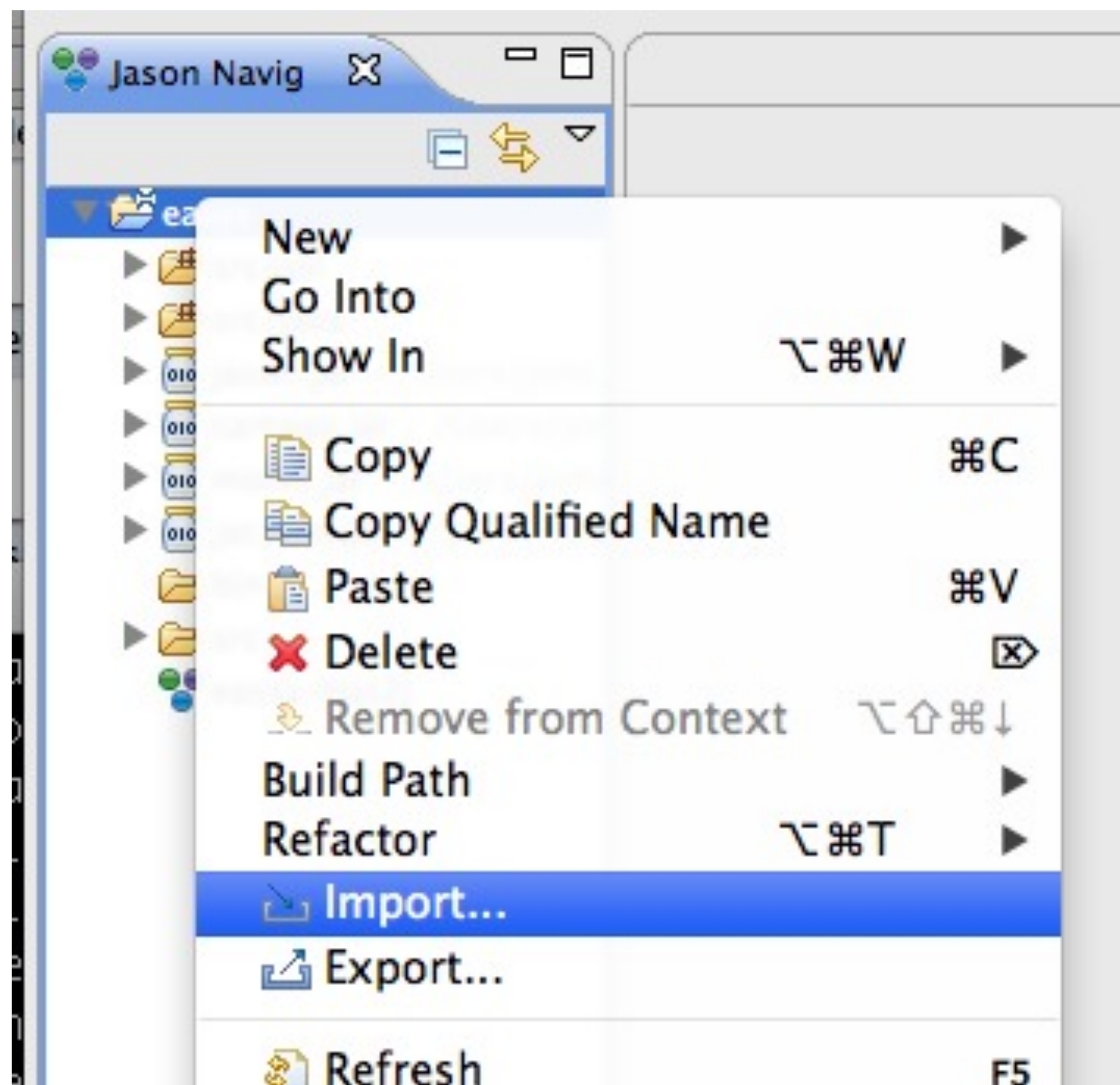
Project name: easss

Infrastructure: JaCaMo

Environment Without environment

? < Back Next > Cancel Finish







Import

Select

Import resources from the local file system into an existing project.



Select an import source:

- ▼ General
 - Archive File
 - Existing Projects into Workspace
 - File System**
 - Preferences
- ▶ CVS
- ▶ Git
- ▶ Install
- ▶ Jason
- ▶ Maven
- ▶ Plug-in Development
- ▶ Run/Debug
- ▶ Tasks
- ▶ Team
- ▶ XML



< Back

Next >

Cancel

Finish

Import

File system

Import resources from the local file system.



From directory: /Users/jomi/workspace/svn-jacamo/dist/build-house-0.3

Browse...

build-house-0.3

- ☐ .settings
- ☐ doc
- ☒ src
 - ☒ asl
 - ☒ java

- ☐ .classpath
- ☐ .project
- ☒ house-map-tut.mas2j
- ☒ logging.properties
- ☐ readme.txt
- ☐ scenario-exercises.pdf

Filter Types...

Select All

Deselect All

Into folder: easss

Browse...

Options

- ☐ Overwrite existing resources without warning
- ☐ Create top-level folder

Advanced >>



< Back

Next >

Cancel

Finish



Jason Navig

- src/asl
 - common.asl
 - companyA.asl
 - companyB.asl
 - companyC.asl
 - companyD.asl
 - companyE.asl
 - giacomo.asl
 - org_code.asl
 - org_goals.asl
 - sample_agent.asl
- src/java
 - simulator
 - Doors.java
 - ElectricalSystem.ji
 - ExteriorPainting.ji
 - Floor.java
 - House.java
 - HousePart.java
 - InteriorPainting.ja
 - Plumbing.java
 - Roof.java
 - Site.java
 - Walls.java
 - Windows.java
 - tools
 - AuctionArt.java
- jason.jar - /Users/jomi/
- cartago.jar - /Users/jor

```

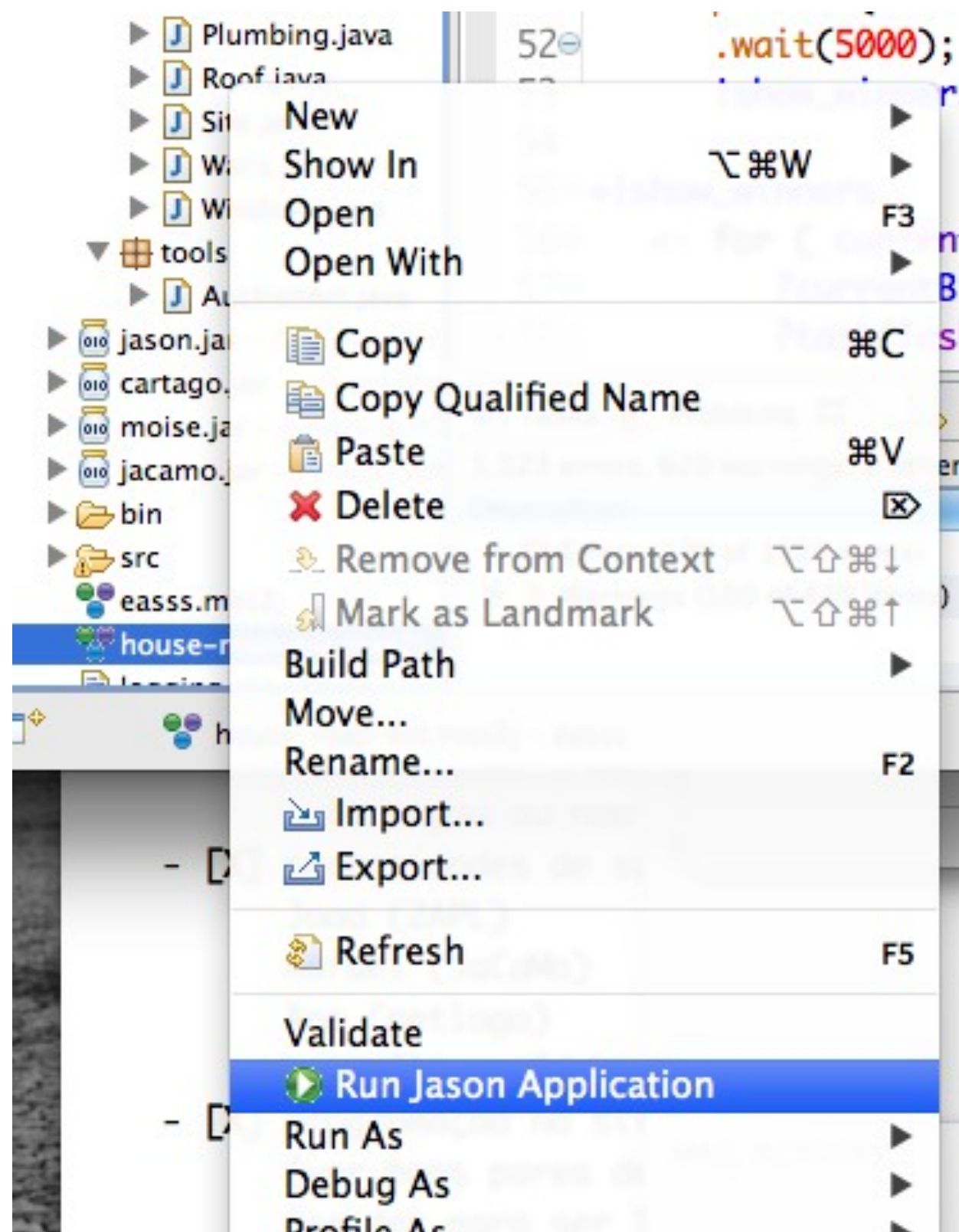
32
33+!create_auction_artifacts
34<- !create_auction_artifact("SitePreparation", 2000);
35!create_auction_artifact("Floors", 1000);
36!create_auction_artifact("Walls", 1000);
37!create_auction_artifact("Roof", 2000);
38!create_auction_artifact("WindowsDoors", 2500);
39!create_auction_artifact("Plumbing", 500);
40!create_auction_artifact("ElectricalSystem", 500);
41!create_auction_artifact("Painting", 1200);
42
43+!create_auction_artifact(Task,MaxPrice)
44<- .concat("auction_for_",Task,ArtName);
45makeArtifact(ArtName, "tools.AuctionArt", [Task, Ma
46focus(ArtId).
47-!create_auction_artifact(Task,MaxPrice)[error_code(Code)
48<- .print("Error creating artifact ", Code).
49
50+!wait_for_bids
51<- println("Waiting the bids for 5 seconds...");
52.wait(5000); // use an internal deadline of 5 secur
53!show_winners.
54
55+!show_winners
56<- for ( currentWinner(Ag)[artifact_id(ArtId)] ) {
57?currentBid(Price)[artifact_id(ArtId)]; // check
?task(Task)[artifact_id(ArtId)]; // and

```

Outline

- number_of_tasks(NS)
- +!have_a_house
- !have_a_house[error(E),error
- +!contract
- +!create_auction_artifacts
- +!create_auction_artifact(Ta
- !create_auction_artifact(Ta
- +!wait_for_bids
- +!show_winners
- +!execute
- +!contract_winners(GroupN
- +!contract_winners(_)
- +?formationStatus(ok)[artifa
- +!house_built

running the application



Plumbing OK

Electrical System OK

Exterior Painting OK

Interior Paining OK

Roof OK

Walls OK

Windows OK

Doors OK

Floor OK

Site OK

src

asl

java

house-os.xml

easss.mas2j

house-map-tut.mas2j

logging.properties

house-map-tut.

MAS Console - house_building

[giacomo] Waiting the bids for 5 seconds...

[giacomo] Winner of task WindowsDoors is companyD8 for 1221

[giacomo] Winner of task SitePreparation is companyD5 for 1055

[giacomo] Winner of task ElectricalSystem is companyC2 for 300

[giacomo] Winner of task Painting is companyC2 for 1100

[giacomo] Winner of task Plumbing is companyA for 300

... ORA4MAS Artifacts GUI ...

... Scheme Board bhsch (build_house_sch) ...

organisation entity

normative state

normative facts

normative program

specification

OK

Responsible groups: [hsh_group](#).

Players

- companyA committed to [install_plumbing](#)
- companyC2 committed to [paint_house](#)
- companyC2 committed to [install_electrical_system](#)
- companyD5 committed to [prepare_site](#)
- companyD8 committed to [fit_doors](#)
- companyD8 committed to [fit_windows](#)
- companyE committed to [build_walls](#)
- companyE committed to [build_roof](#)
- companyE committed to [lay_floors](#)
- giacomo committed to [management_of_house_building](#)

goal	state	committed/achieved by	arguments	plan
house_built	satisfied	[giacomo]/[giacomo]		= site_prepared , floors_laid , walls_built , rwd , pee , interior _painted
site_prepared	satisfied	[companyD5]/[companyD5]		
floors_laid	satisfied	[companyE]/[companyE]		
walls_built	satisfied	[companyE]/[companyE]		
rwd	satisfied	[]/[]		= roof_built windows_fitted doors_fitted

Exercises

1. Change the code of the auction artifact to:

- (a) create a new observable property that shows the state of the auction (open or closed)
- (b) add a new operation `clearAuction` (after `clearAuction`, the state of the auction becomes closed and attempts to use the `bid` operation will fail)

2. Change the house owner program so that the agent uses the new `clearAuction` operation

Homework

1. Change the auction artifact so that it shows and manages the bidding deadline and has a new operation for starting the auction; the `clearAuction` operation is no longer needed
2. Create a new company for one of the tasks and give it any bidding strategy you like

Homework

3. Choose a new auction mechanism and implement a new auction artifact that implements that mechanism; you should not change the agents for this exercise
4. Now choose another mechanism that will require different strategies in the agents and implement them

Moise Specification

Building Phase

- After all auctions are over, Giacomo sends messages for the hired companies to enter into the execution phase
- A virtual organisation is created to assist with coordination and cooperation in the execution of the global workflow
- Implementation choice:
 - The organisation is specified using the Moise organisation modelling language

Building: Organisation Side

- Moise functional specification is used to define the workflow
- Moise structural specification is used to define the role and group structures
- Moise normative specification is used to distribute the tasks of the workflow to the roles

Functional Specification

- The functional specification simply defines a social scheme for the global workflow

$a ; b ; c ; (d \mid e \mid f) ; (g \mid h \mid i) ; j$

- One **mission** for each task except for the painting of the exterior and of the interior of the house that are grouped into the same mission
- A task for the management of the execution of the workflow is also added

Functional Specification

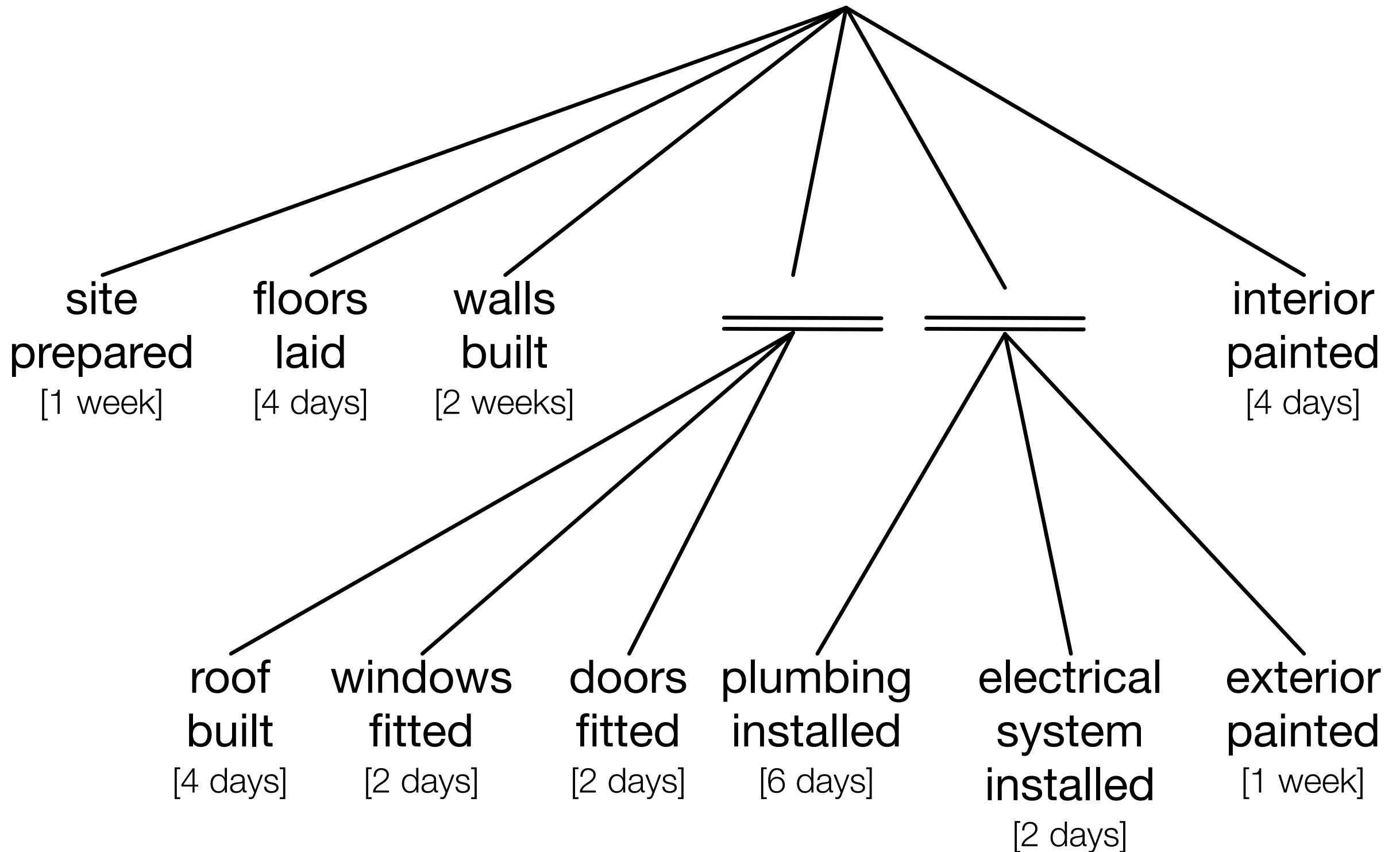
- The functional specification simply defines a social scheme for the global workflow

a ; b ; c ; (d | e | f) ; (g | h | i)

- One **mission** for each task except painting of the exterior and of the interior of the house that are grouped into a mission
- A task for the management of the workflow is also added

- (a) Site preparation
- (b) Lay floors
- (c) Build walls
- (d) Build roof
- (e) Fit windows
- (f) Fit doors
- (g) Install the plumbing
- (h) Install the electrical system
- (i) Paint the exterior of the house
- (j) Paint the interior of the house

house built



Structural Specification (I)

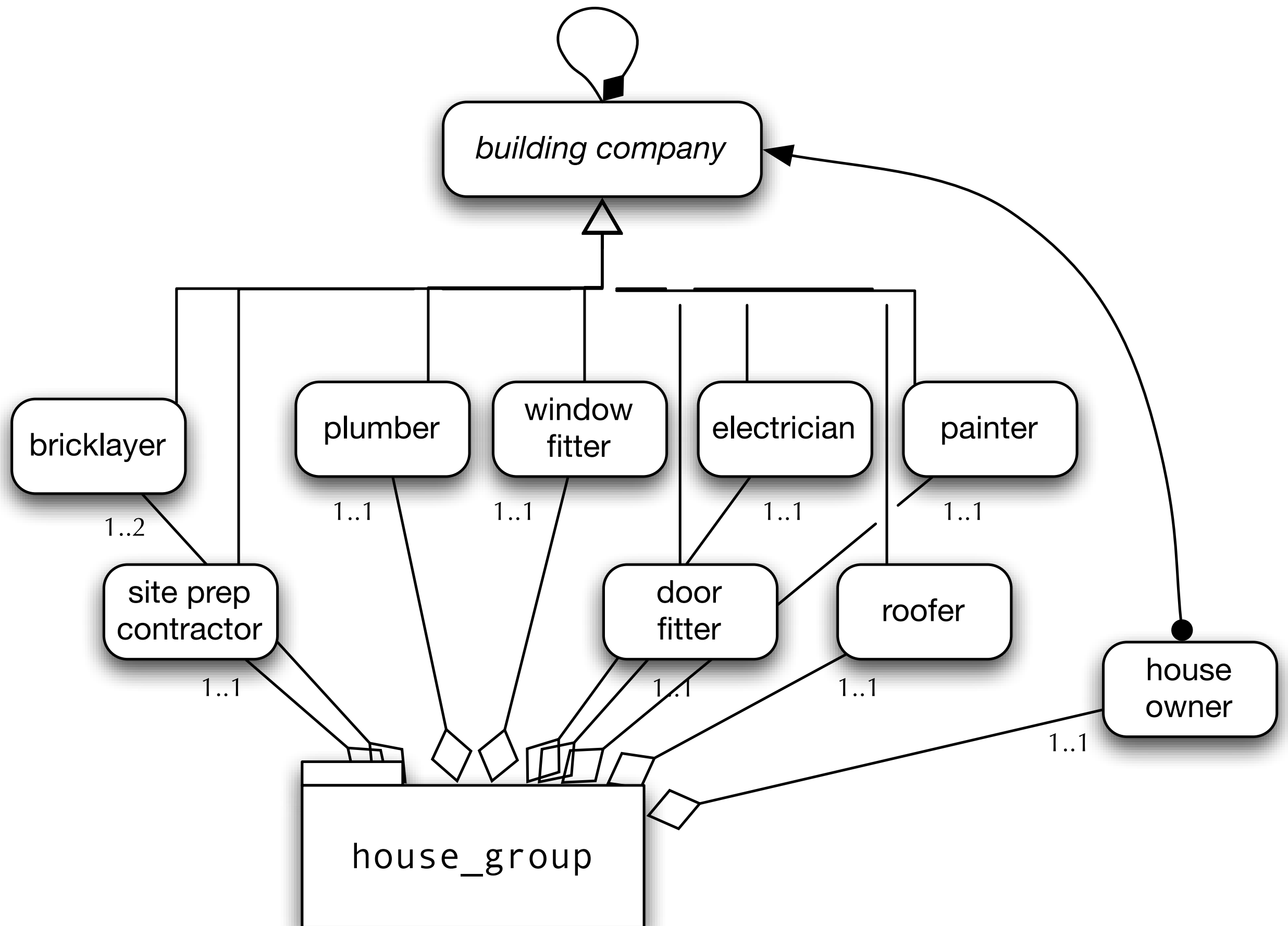
- Role hierarchy
 - `house_owner`
 - `building_company`
 - abstract role, specialised into:
`site_prep_contractor,`
`bricklayer, roofer,`
`window_fitter, door_fitter,`
`plumber, electrician, painter`

Structural Specification (2)

- The roles are used in a **group** called 'house_group' where:
 - house_owner has cardinality (1,1)
 - site_prep_contractor has cardinality (1,1)
 - bricklayer has cardinality (1,2)
 - roofer has cardinality (1,1)
 - window_fitter has cardinality (1,1)
 - door_fitter has cardinality (1,1)
 - plumber has cardinality (1,1)
 - electrician has cardinality (1,1)
 - painter has cardinality (1,1)

Structural Specification (3)

- Notes:
 - the role `building_company` is compatible with `building_company` so that the same agent can play more subroles
 - role `house_owner` has authority over the `building_company` role
 - a communication link connects the role `build_company` to `house_owner`



Normative Specification (I)

- Given the definition of the **missions**, the following **norms** are defined:
 - any agent playing the **role** `houseOwner` is **obliged to commit to mission** `'mManagement'`
 - `role site_prep_contractor` to the mission concerning the site preparation goal
 - `role bricklayer` to the mission of laying the floors
 - `bricklayer` is also obliged to commit to the mission of building the walls
 - `roofer` is obliged to the mission of building the roof

Normative Specification (2)

- `role window_fitter` to the mission related to fitting the windows
- `door_fitter` is obliged to commit to the mission of fitting the doors
- `plumber` to installing the plumbing
- `electrician` to installing the electrical system
- `painter` to the mission concerning the painting of the house

Building: Agent Side

- Owner agent is equipped with the plans to construct the virtual organisation based on the result of contracting phase
- Company agents have plans to enter the organisation, adopt the role corresponding to their contract and to catch the different events generated by the OMI
- Companies have plans to execute autonomously the various actions related to the goals related to the missions they are committed to in the organisation scheme
- NB: agents are benevolent with respect to the organisation, i.e. they don't violate the norms

Building: Environment Side

- Artifacts that model the state of the environment (e.g., model the state of the construction of a wall)

Exercises

- I. Do the following changes in the organisation specification:
 - (a) tasks `site_preparation` and `lay_floors` can be done in parallel
 - (b) all tasks have to be done in sequence

Homework

1. Develop an agent that tries to adopt roles related to tasks he is not supposed to (malevolent agent!)
(e.g. Giacomo trying to play some company role)
2. Develop an agent that does not fulfil the tasks

Homework

3. Change the Giacomo agent so that it reacts to the norm violation
 - Giacomo should create a new auction for that task and forbid the violating company from taking part in the new auction
4. Change the system to build two houses in parallel

Homework

5. Change the Giacomo agent so that it reads the Moise specification and creates the necessary auction artifacts based on the specified tasks
6. Change Giacomo so that it is able to monitor the building of the house and check whether the tasks are being done appropriately

Conclusions

- Separation of concerns
 - agent, environment, organisation, interaction
 - $MAP = AOP + EOP + OOP + IOP$
- Design == Implementation
- In our code, (almost) no (direct) communication in the MAS!