Practical Session

Multi-Agent Oriented Programming

The JaCaMo Platform

Web Intelligence Master 2012

Olivier Boissier

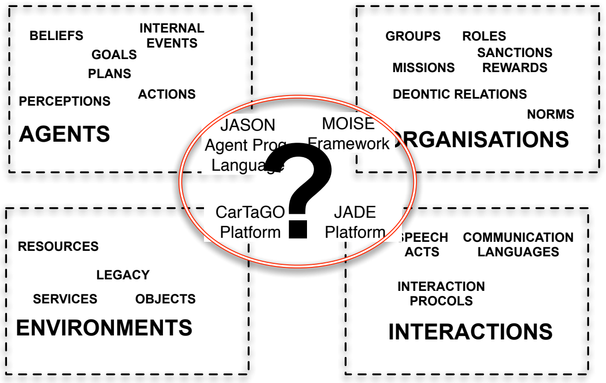
ENS Mines Saint-Etienne
http://www.emse.fr/~boissier

Web Intelligence Master — Nov 2012

from EASSS 2012, European Agent Systems Summer School, May, 2012 @ Valencia, Spain
Thanks to R. Bordini, J.F. Hubner, A. Ricci

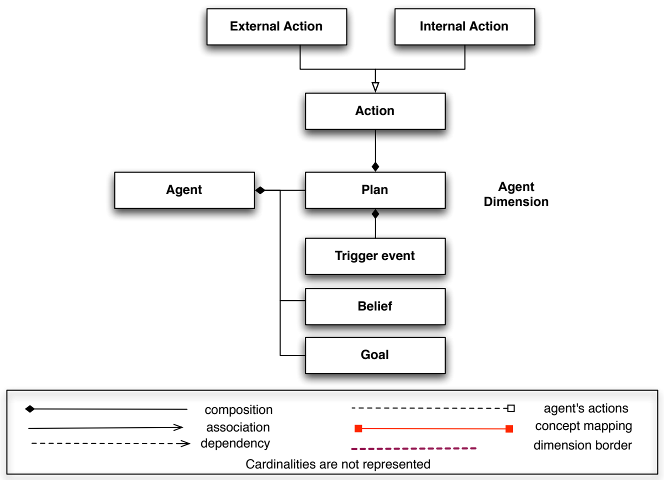
Outline

Putting the Pieces Together

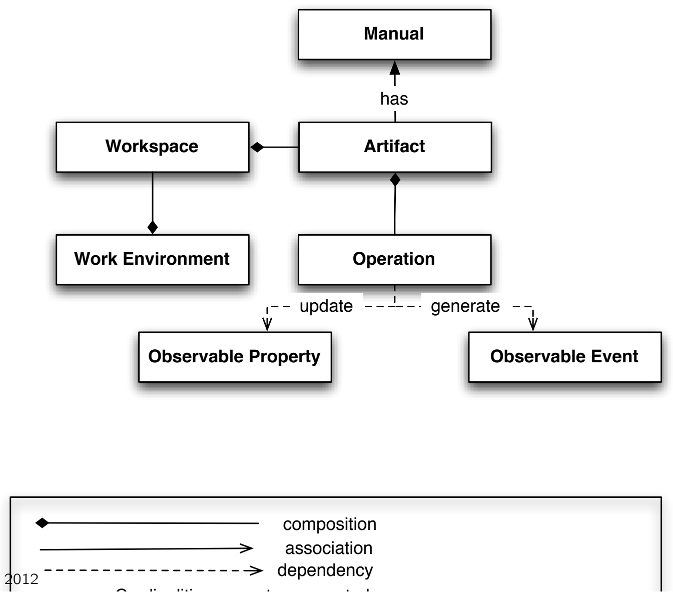


⇝ The *JaCaMo* Platform

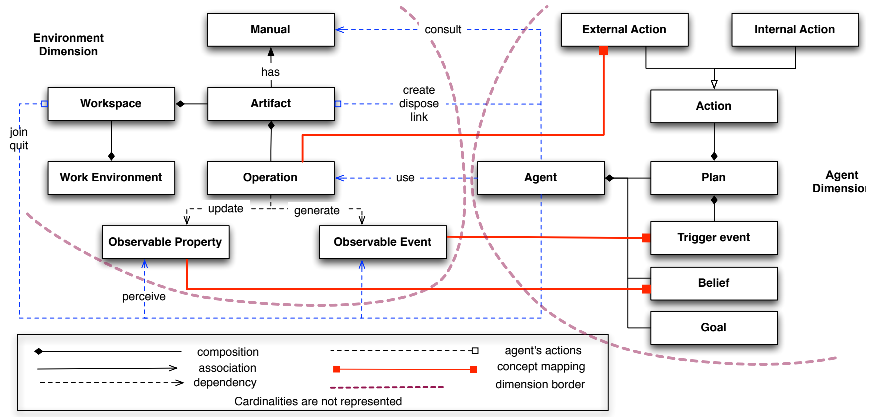
Agent meta-model



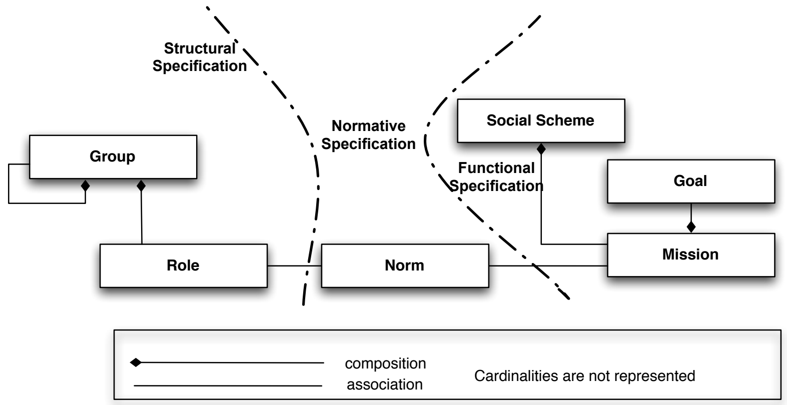
Environment meta-model



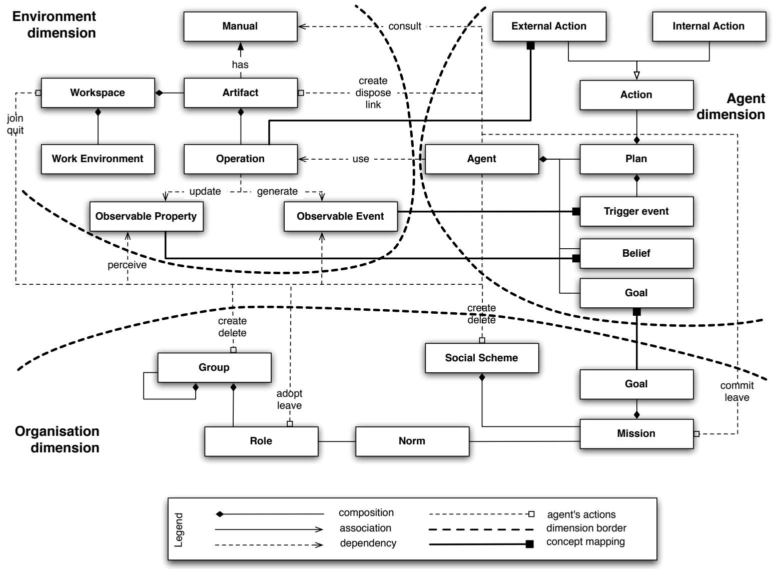
A & E Interaction meta-model



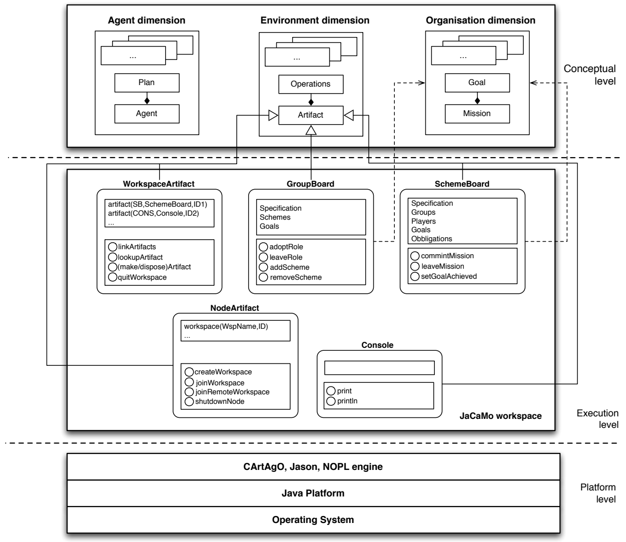
Organisation meta-model



JaCaMo Meta-Model



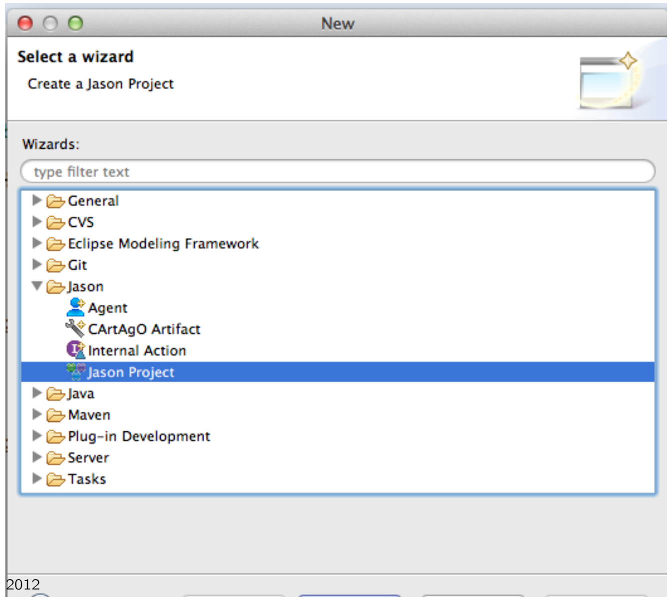
JaCaMo Platform



Downloads

- *JaCaMo* within the eclipse platform
  - *Jason* $\geq$ 1.3.7 `https://jason.sf.net`
  - Jason Eclipse Plugin
    `https://docs.google.com/document/pub?id=1URDYMtFP64rHnlb7GcnKyUGbaypNcZmteWupWD5p6sM`
- Other Relevant URLs and Projects
  - *JaCaMo* `http://jacamo.sourceforge.net`
    - *Jason* `http://jason.sourceforge.net`
    - CArtAgO `http://cartago.sourceforge.net`
    - *Moise* `http://moise.sourceforge.net`
  - `http://jaca-android.sourceforge.net`
  - `http://jaca-web.sourceforge.net`
  - JaCaMo-Arduino (see JaCaMo website)
  - `http://jason.sourceforge.net/jBook`

JaCaMo Wizard


New

Select a wizard

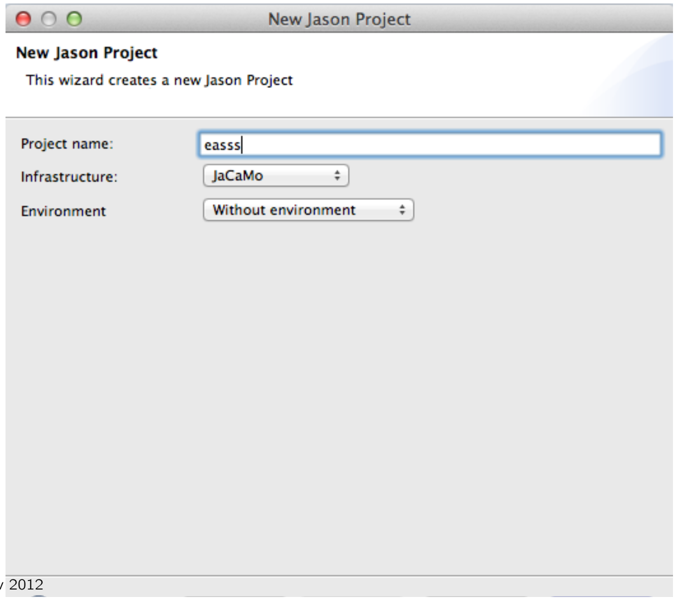
Create a Jason Project

Wizards:

type filter text

- General
- CVS
- Eclipse Modeling Framework
- Git
- Jason
  - Agent
  - CArtAgO Artifact
  - Internal Action
  - Jason Project
- Java
- Maven
- Plug-in Development
- Server
- Tasks

Configuring the Infrastructure

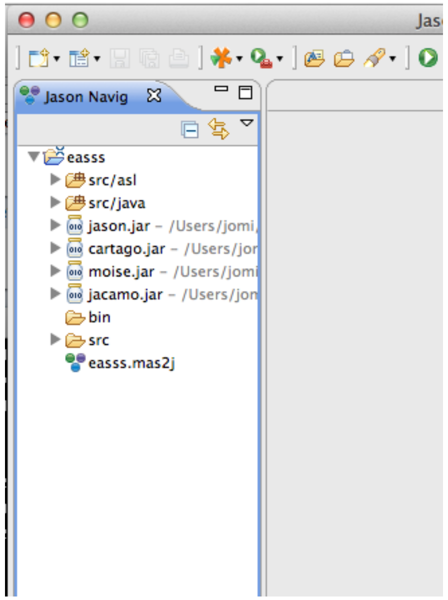

New Jason Project

New Jason Project

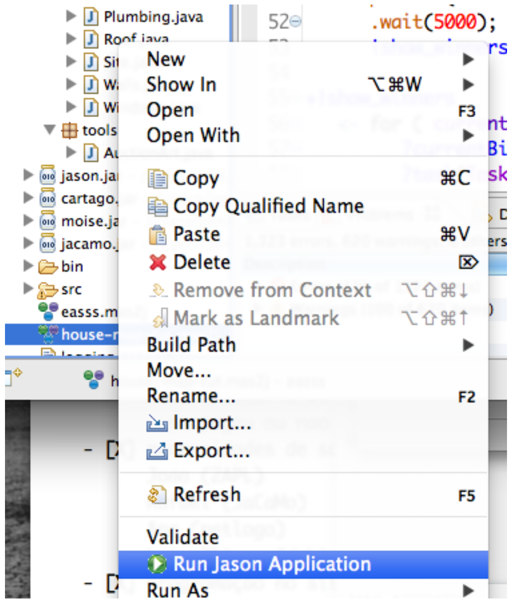
This wizard creates a new Jason Project

Project name:	easss
Infrastructure:	JaCaMo
Environment	Without environment

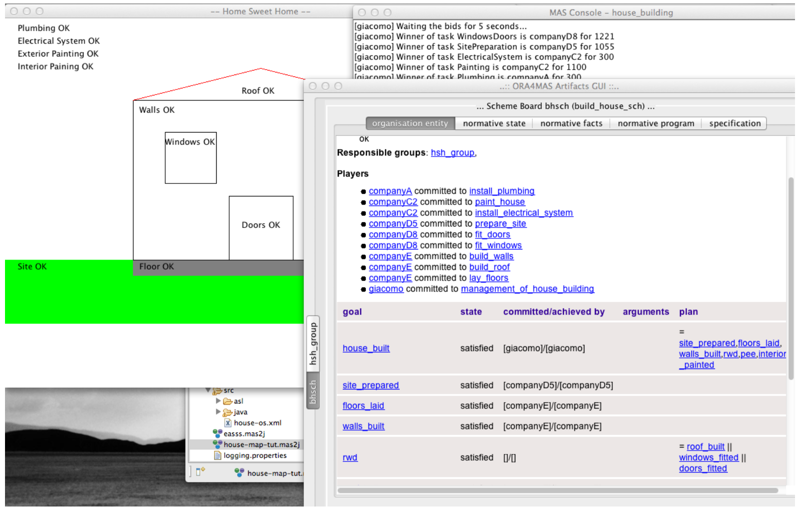
Code organization



Running an *JaCaMo* Application



Application in execution



```
[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
```

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

Outline

1. The JaCaMo Multi-Agent Oriented Programming Platform
   - Global view
   - Setting-up the JaCaMo platform
2. Building House Use Case
   - Scenario
   - Multi-Agent Oriented Modelling
     - Contracting Phase
     - Building Phase
   - Multi-Agent Oriented Programming
     - Contracting Phase
     - Building Phase

Context

- Giacomo, a CyberWorld addict, wants to build a house
- Two main phases are considered:
  1. Contracting specialised companies
     - Giacomo hires various companies specialised in different aspects of house construction.
     - The companies are independant and want to keep their autonomy of decision.
  2. Building of the the house
     - Contracted companies execute the main workflow for building the house
     - Giacomo supervises the execution of the workflow

Contracting specialised companies (Phase 1)

- Tasks for which companies should be contracted:

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

- The same company can be hired for more than one task

Building the house (Phase 2)

- Once companies have been contracted, 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).

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

Where: (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

Objective

- The objective of this practical work is to use the different abstract dimensions participating in the definition of Multi-Agent Oriented Programming to program a Multi-Agent System that could support the execution of this scenario.
- Given the constrainted framework of a course, this complex application is simplified while keeping the following features:
  - companies and Giacomo delegated part of their decisions to agents: one or several agents for Giacomo, one or several agents for each company
  - agents attached to Giacomo or Company are autonomous: no central control is possible
  - open system (companies may leave or enter the system at any moment) and dynamic environment

Modelling common to both phases

- House Owner Agent (Giacomo's assistant):
  - provides the requirements for the house, with budget limitations
  - helps Giacomo in the different decisions: contracting companies, managing and supervising the workflow execution
- Company Agent:
  - offer the house building services, skills and resources of the company
  - if hired, execute the house building tasks

Modelling Contracting Phase

- Interaction Modelling: since the different companies aim at keeping their autonomy, choice of a negotiation protocol in order to build contracts between Giacomo and the different companies for each of the tasks
- Organisation Modelling: flat organisation is defined with no particular role
- Environment Modelling: no particular environment concerns are considered in this phase

Interactions

- Use of Electronic Auctions 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

Modelling Building Phase

- Interaction Modelling: At the end of the contracting phase, Giacomo contact the hired companies to enter into the execution phase and enact the negotiated workflow
- Organisation Modelling: a virtual organisation is created to assist with coordination and cooperation in the execution of the global workflow
- Environment Modelling: Graphical Interface to show the global state of the building

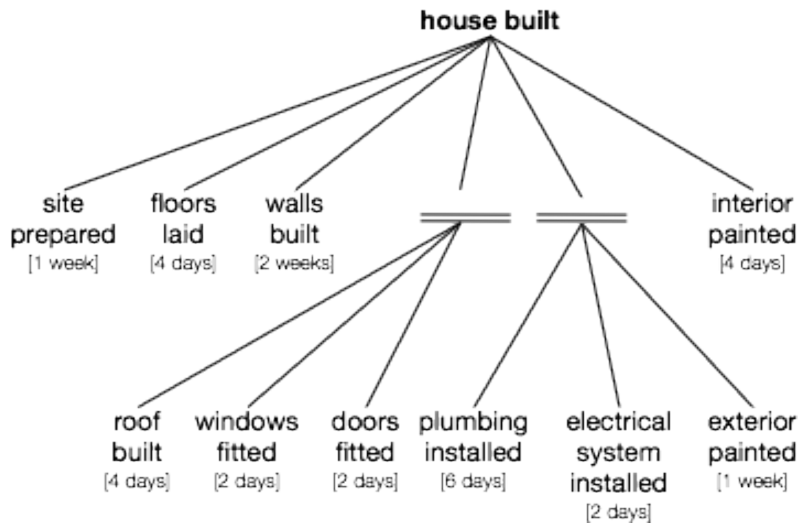
Organisation

- $\mathcal{M}$oise functional specification is used to define the workflow
- $\mathcal{M}$oise structural specification is used to define the role and group structures
- $\mathcal{M}$oise 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
- $a; b; c; (d|e|f); (g|h|i); j$
- 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

Example: Organisation Functional Specification



Structural Specification

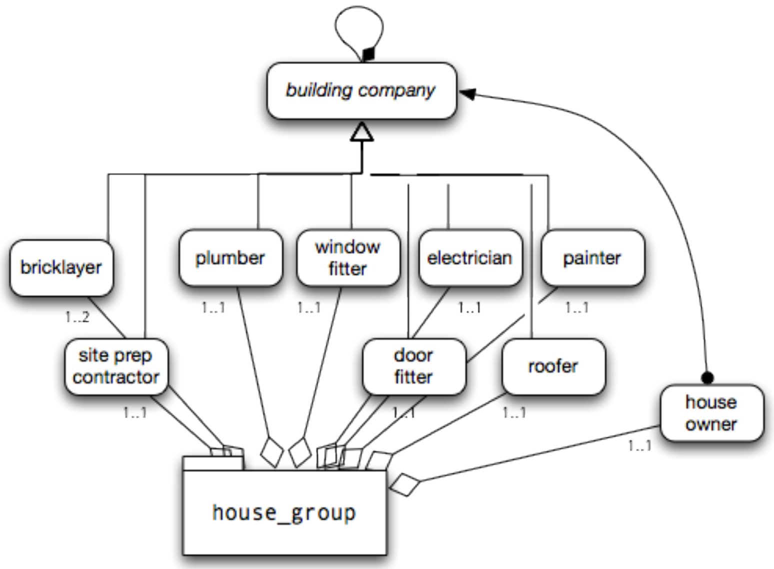
- Roles: `house_owner`, `building_company` (abstract role), `site_prep_contractor`, `bricklayer`, `roofer`, `window_fitter`, `door_fitter`, `plumber`, `electrician`, `painter`
- Inheritance Hierarchy: `building_company` abstract role, specialised into: `site_prep_contractor`, `bricklayer`, `roofer`, `window_fitter`, `door_fitter`, `plumber`, `electrician`, `painter`
- Group: roles are used in a `house_group` group

Structural Specification

In group house_group,

- cardinalities are:
  - (1,1) for house_owner, site_prep_contractor, roofer, window_fitter, door_fitter, plumber, electrician, painter
  - (1,2) for bricklayer ;
- the same agent can play more subroles ⇝ the role building_company is compatible with building_company
- role house_owner has authority over the building_company role
- a communication link connects the role build_company to house_owner

Example: Organisation Structural Specification



Example: Organisation Normative Specification

norm	modality	role	mission / goals
n1	O	house_owner	house built
n2	O	site_prep_contractor	site prepared
n3	O	bricklayer	floors laid, walls built
n4	O	roofer	roof built
n5	O	window_fitter	windows fitted
n6	O	door_fitter	doors fitted
n7	O	plumber	plumbing installed
n8	O	electrician	electrical system installed
n9	O	painter	interior painted, exterior paint

Outline

Main files

- house-map-tut.mas2j: project
- src/asl: agent code
- src/java: environment code
- src/house-os.xml: organisation code
- Initial project
  `http://sourceforge.net/projects/jacamo/files`

Programming Agents

- The House owner agent is programmed in *Jason*
- The Company Agents are programmed in *Jason* (or 2APL, Jadex, ...) $\rightsquigarrow$ Heterogeneous Agent System

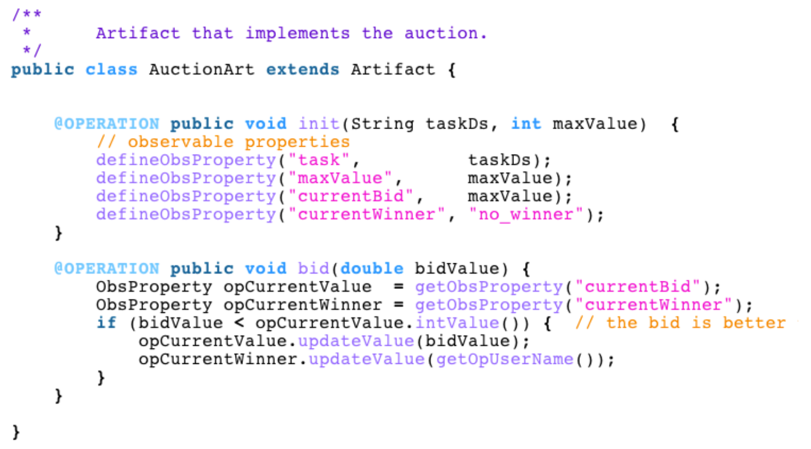
Programming Interaction in the Environment

- Encapsulation of the auction mechanism in an auction artifact
- 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

Auction Artifact

- observable properties:
  - task description
  - maximum payment value
  - current best bid (lower service price)
  - 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

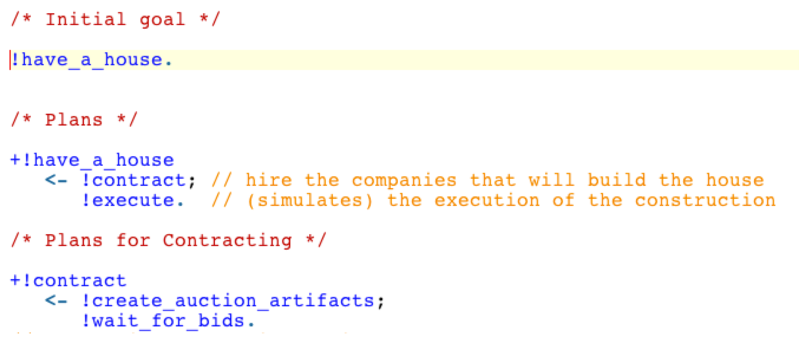


```
/**
 *      Artifact that implements the auction.
 */
public class AuctionArt extends Artifact {


    @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());
        }
    }

}
```

Programming Interaction in the Agent Side

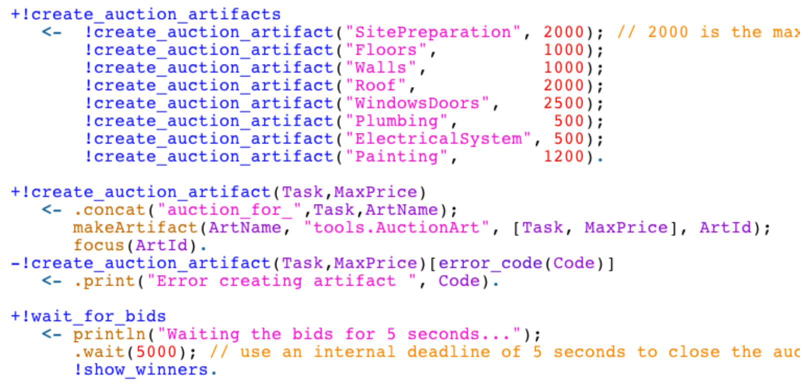
- Giacomo Agent:
  - Plans to launch all auctions by creating the corresponding auction artifacts
  - After some time looks at the best bid in each auction artifact and awards a contract for the winning company
- Company Agents;
  - Plans to look for the auction artifacts of their interest
  - Plans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

src/asl/giacomo.asl



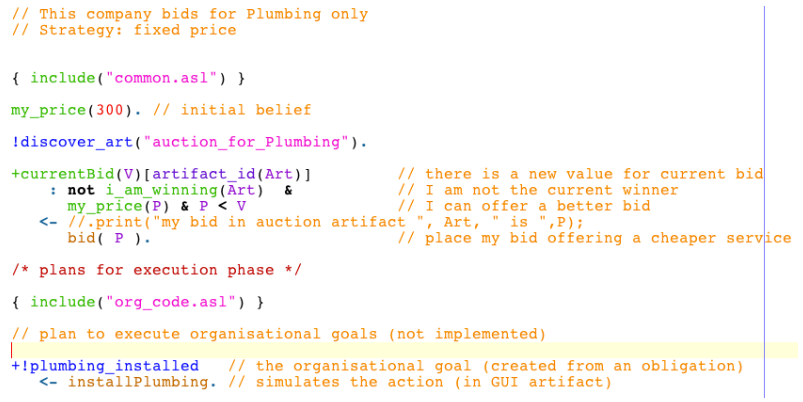
```
/* Initial goal */

!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)
```

Exercises

1. Change the code of the auction artifact to:
   1. create a new observable property that shows the state of the auction (open or closed)
   2. 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
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

Outline

Programming Agents for the Building Phase

- Giacomo Agent is enriched with:
  - Plan to send messages to the hired companies to enter into the execution phase, after all auctions are over.
  - Plan to build the virtual organisation based on the result of the contracting phase: allocating the different roles to the contracted company agents
- Company Agent:
  - Plans to enter the organisation, adopt the role corresponding to their contract and to catch the different events generate by the Organisation
  - 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

Programming Organisation

- 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

Programming Environment

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

Exercises

Do the following changes in the organisation specification:

1. tasks `site_preparation` and `lay_floors` can be done in parallel
2. 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
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
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