"Symbolic" Artificial Intelligence
(*a.k.a. Probabilistic Intelligent Agents*)

INF5033 – S9 - MAJ - IADS 2024-2026

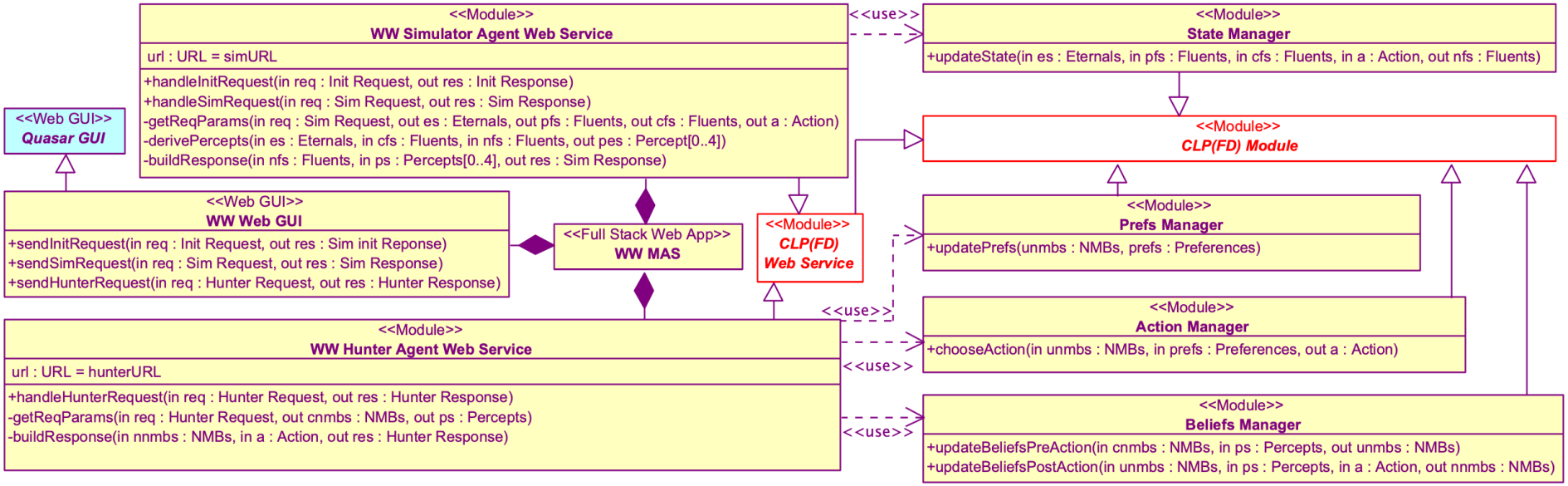
Topic 2b: The Probabilistic Wumpus World Hunter Agent

Jacques Robin, Camilo Correa, Antoine Leblanc © 2022-2026

Section outline

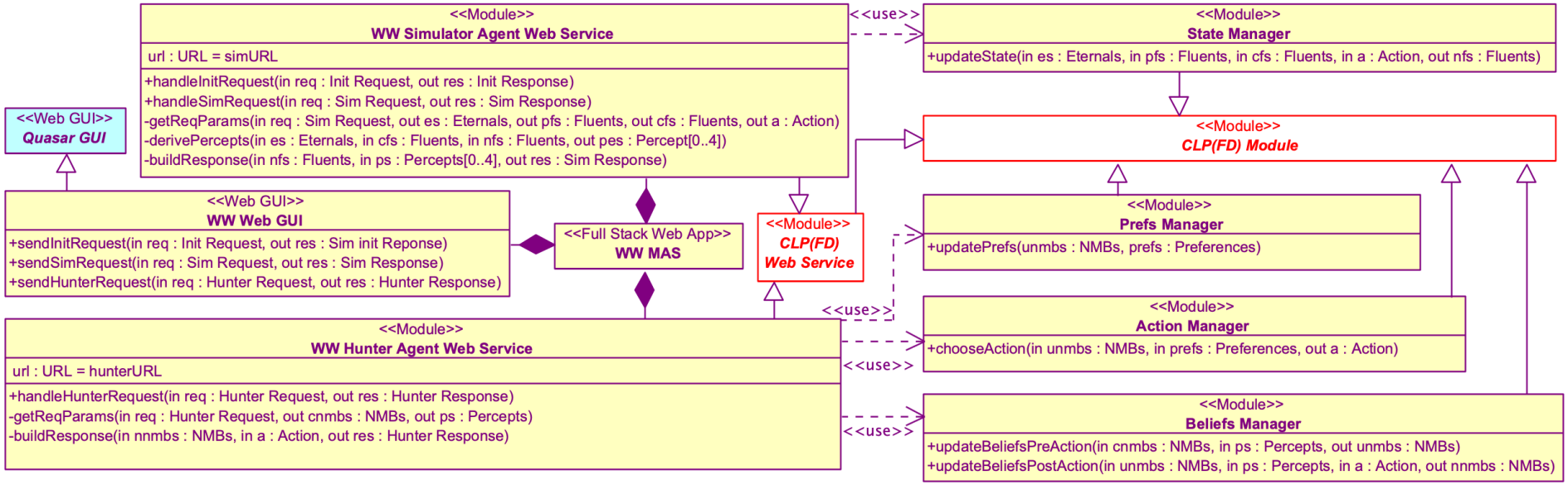
1. The *Wumpus World (WW)* as a *Multi-Agent System (MAS)*
2. UML architecture model
3. HW2 topic
4. Prolog code of WW Simulator

WW MAS Top-Level Architecture



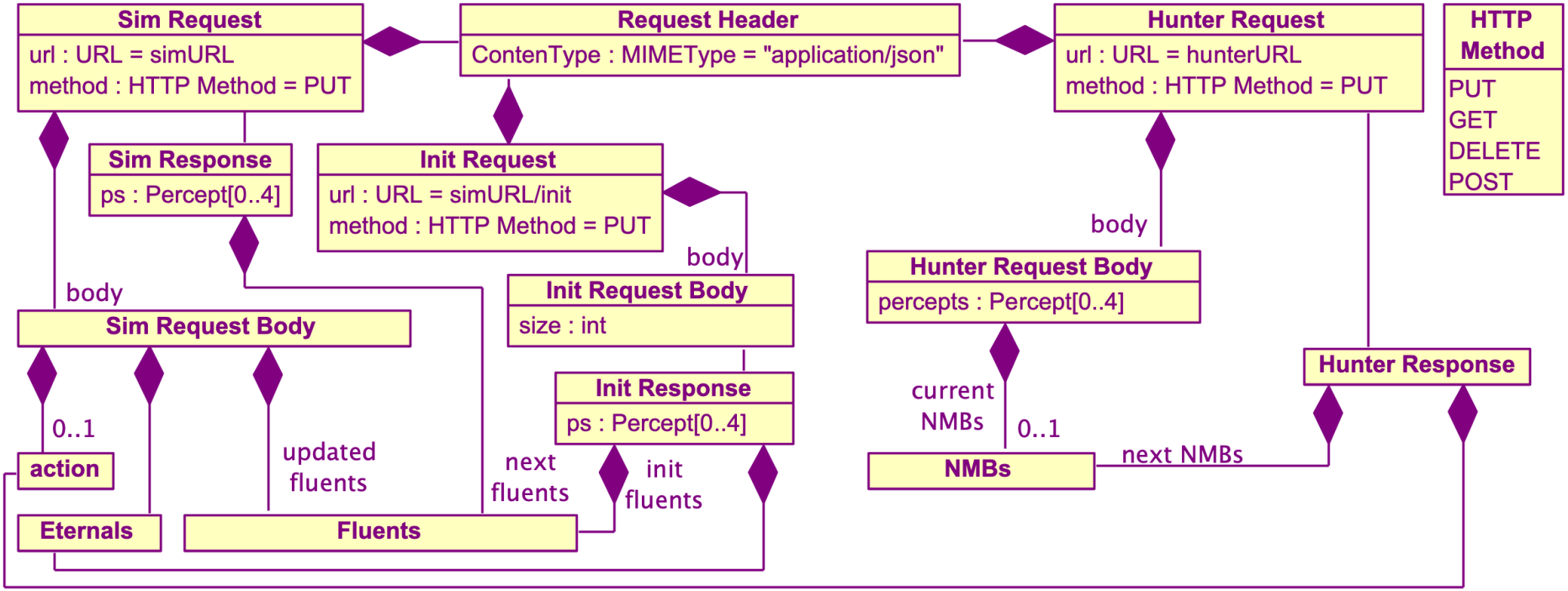
- Minimalist MAS: 2 intelligent agents in mirror image
- One WW gold hunter agent + one WW simulator agent
- Hunter agent:
  - Receives percepts from simulator, updates its beliefs and preferences, choose an action to execute and sends to the simulator agent
- Simulator agent:
  - Its percepts are the hunter agent's action
  - It updates the actual state of the WW
  - Derives from it the percepts of the hunter agent and send them

WW MAS Top-Level Architecture



- Both agents are CLP web services
- To simplify implementing those services and keep as declarative as possible, they are stateless (REST)
- A JavaScript-HTML-CSS GUI client runs in a web browser
- It keeps in the browser local storage both the WW state and the hunter agent's beliefs (partial, uncertain, observed WW substate)
- It is implemented using the JS-HTML-CSS Quasar frontend framework, a layer on top of the Vue framework

WW MAS: HTTP Messages

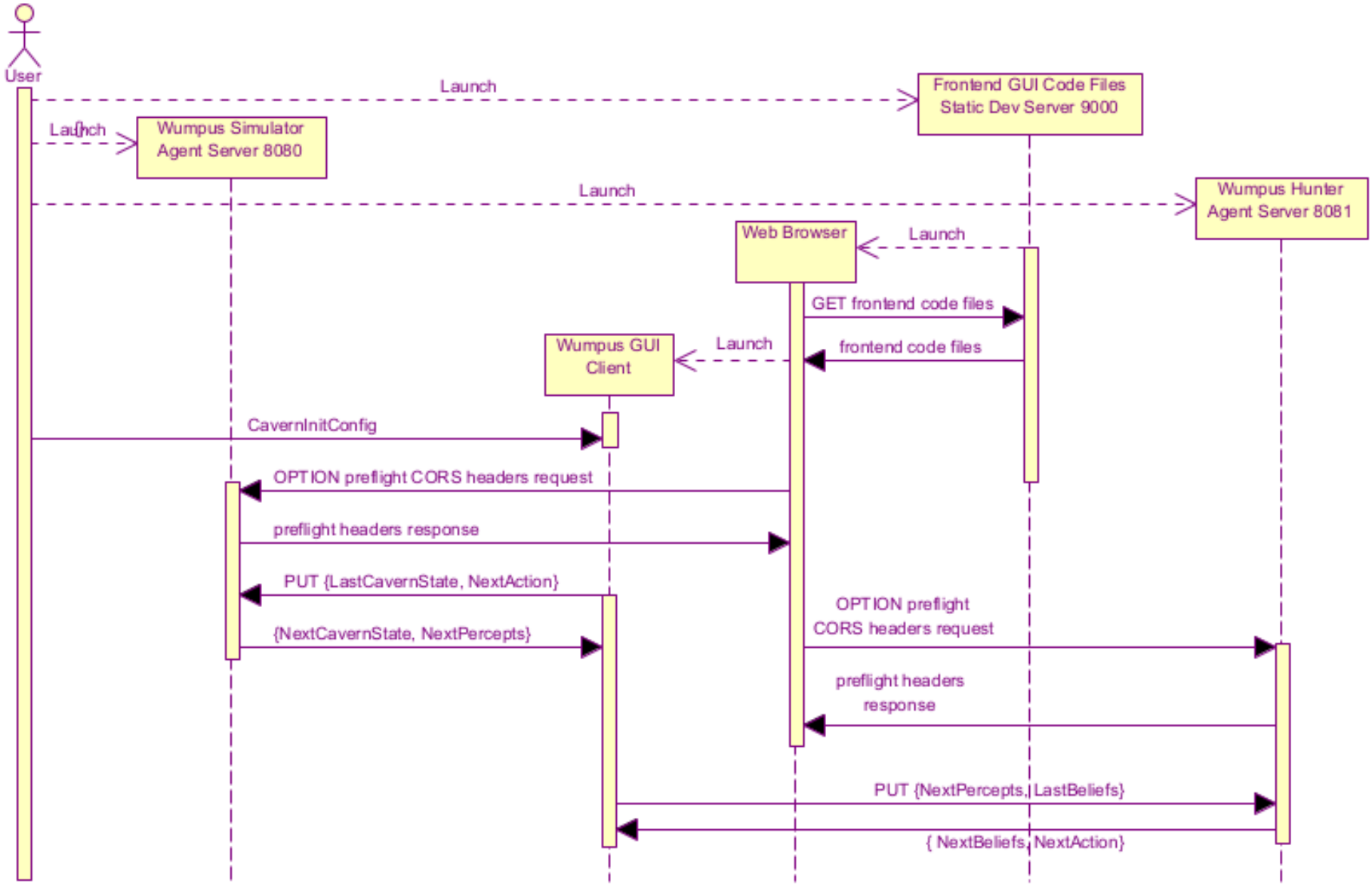


- Since the GUI keeps the state of both agents
- It sends at each game step:
  - To the simulator agent, in addition to the new plan to execute, the WW states that the GUI received from the simulator at the last 2 game steps
  - To the hunter agent, in addition to the new percepts, the WW beliefs and preferences that the GUI received from the hunter at the last 2 game steps
- Eternal WW properties and certain eternal beliefs sent in step 1

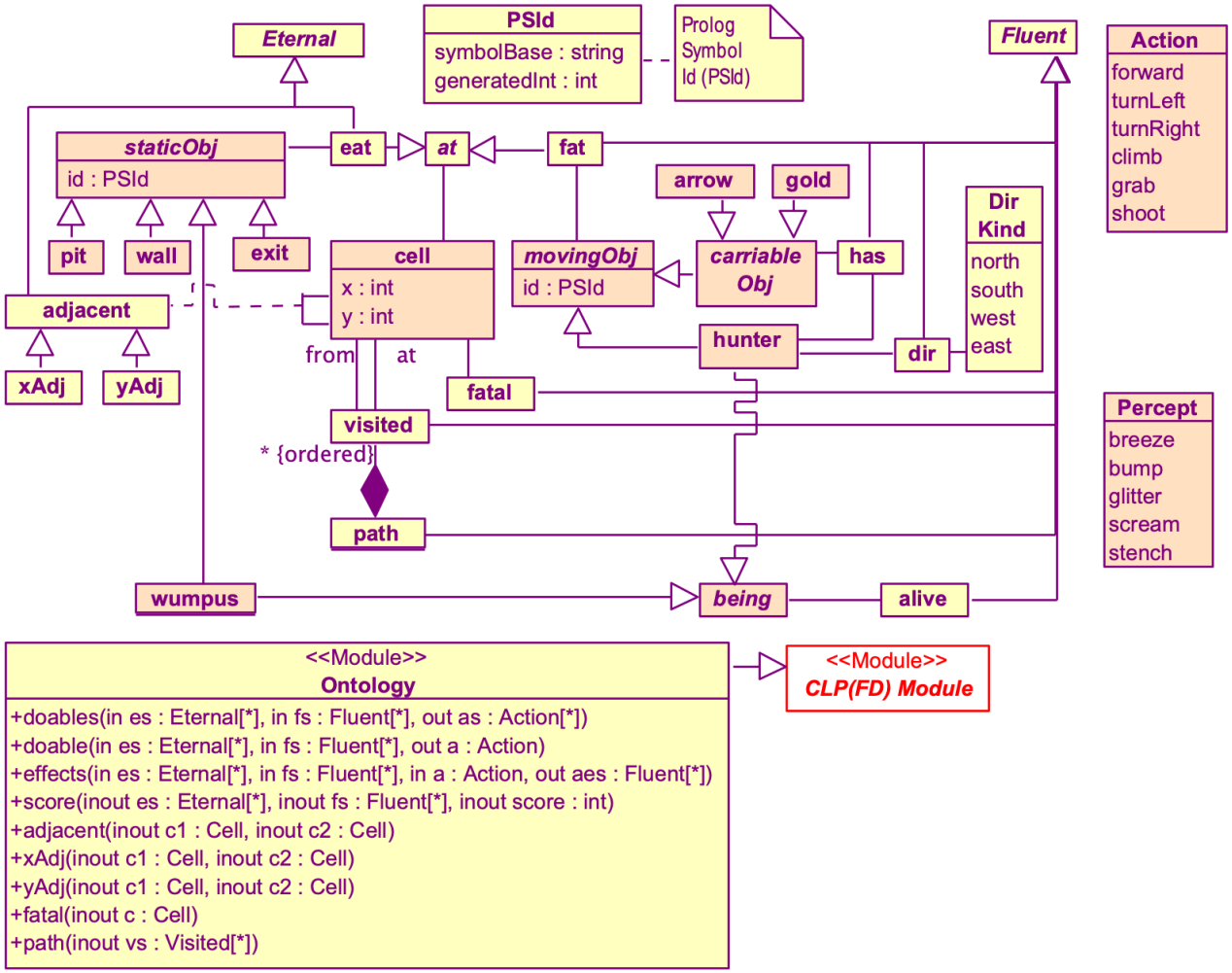
Cross-Origins Resource Sharing (CORS)

- Origin: part of the URL including the protocol, domain name and port, *e.g.*, http://localhost/9000
- When a browser receives the frontend code of a web app from a server at a given origin
- It blocks, for security reasons, the execution of direct requests to another origin
- To circumvent this block:
  - Request by a web app frontend to each server, with an origin that differs from the one from which the frontend app code was served
  - Must be configured to receive OPTION requests with preflight headers
  - And reply with an HTTP response with an empty payload but with headers
  - Allowing the browser to verify that the server accessed
  - Is a legitimate backend for the frontend app that it currently executes
  - Only then does the browser send to the now verified server, the original HTTP request from the frontend app
- Your Wumpus hunter agent web server must be configured to respond to those OPTION request with preflight headers
- Simply copy and adjust such configuration code from the Wumpus simulator agent web server

Interaction GUI SimServer HunterServer

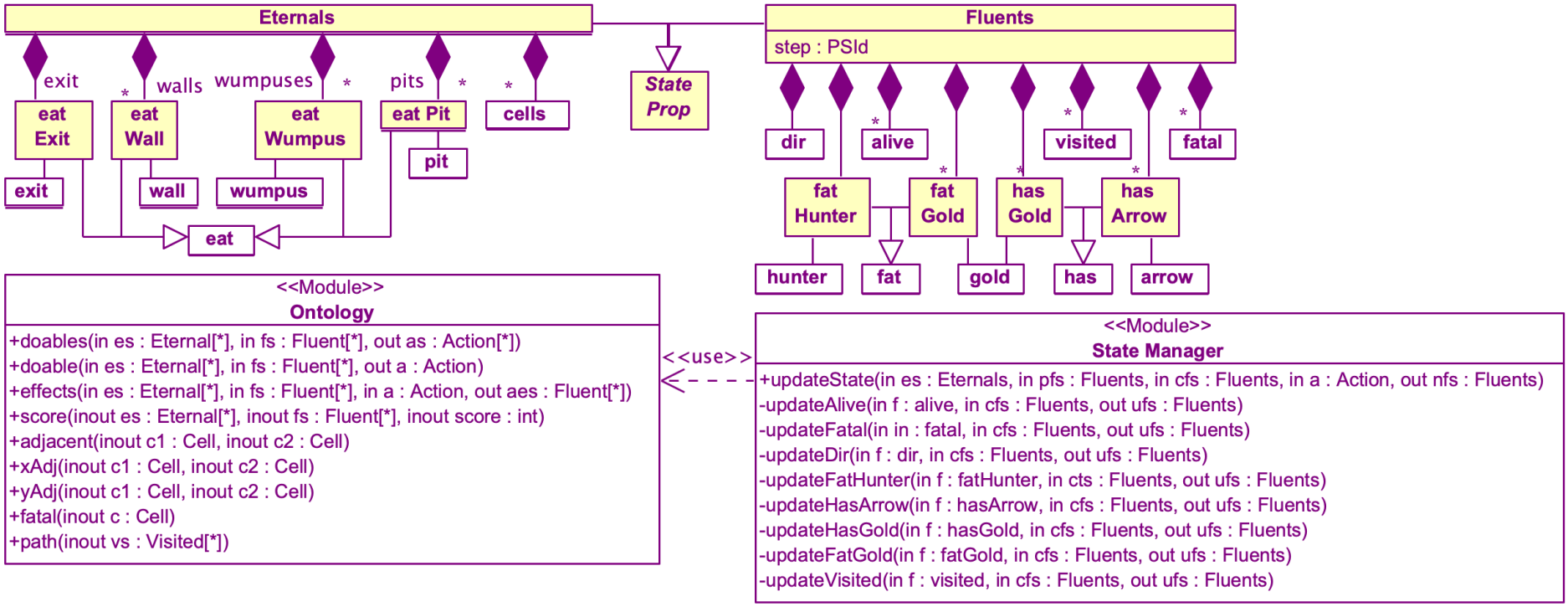


WW Ontology



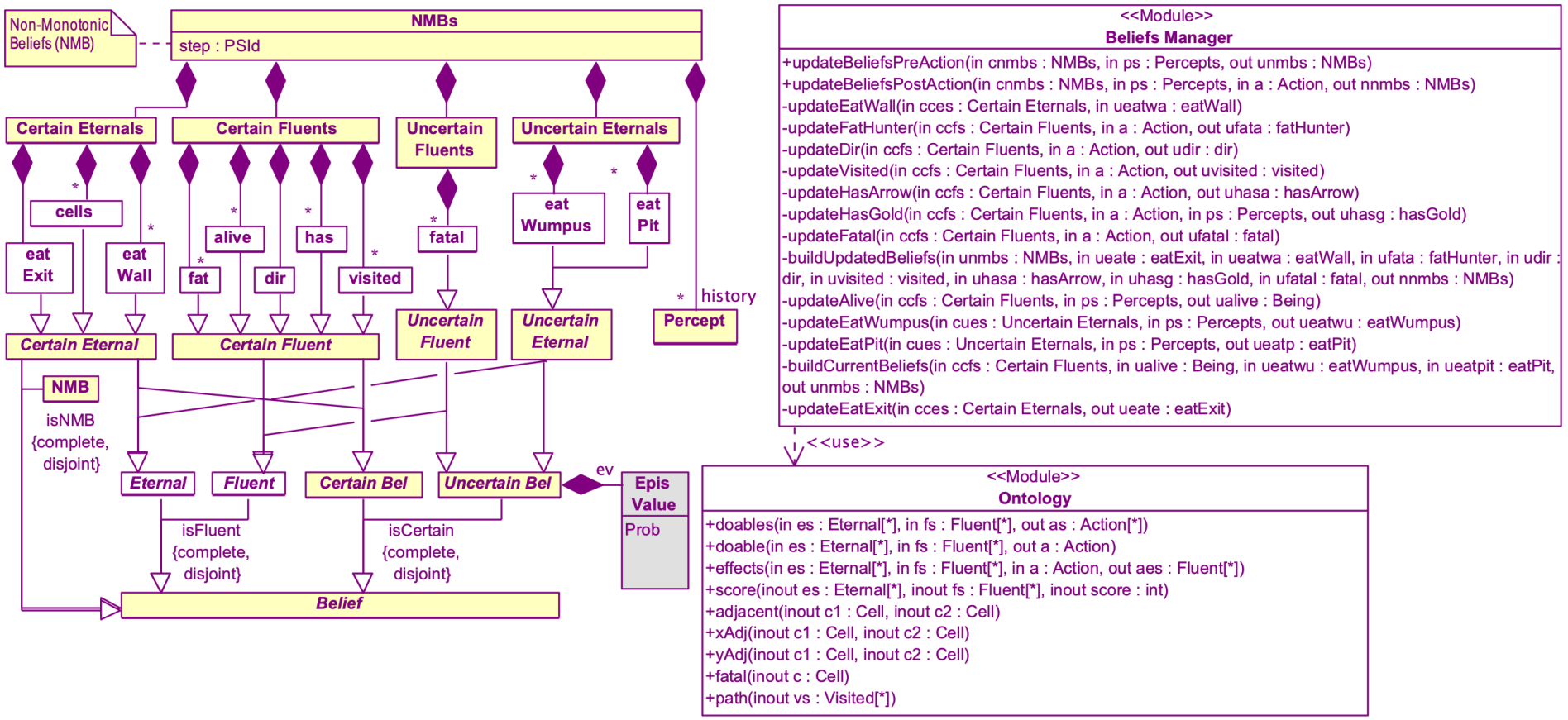
- Common concepts shared by both agents
- Reused in the simulator agent's state and the hunter agent's beliefs

WW (actual) state



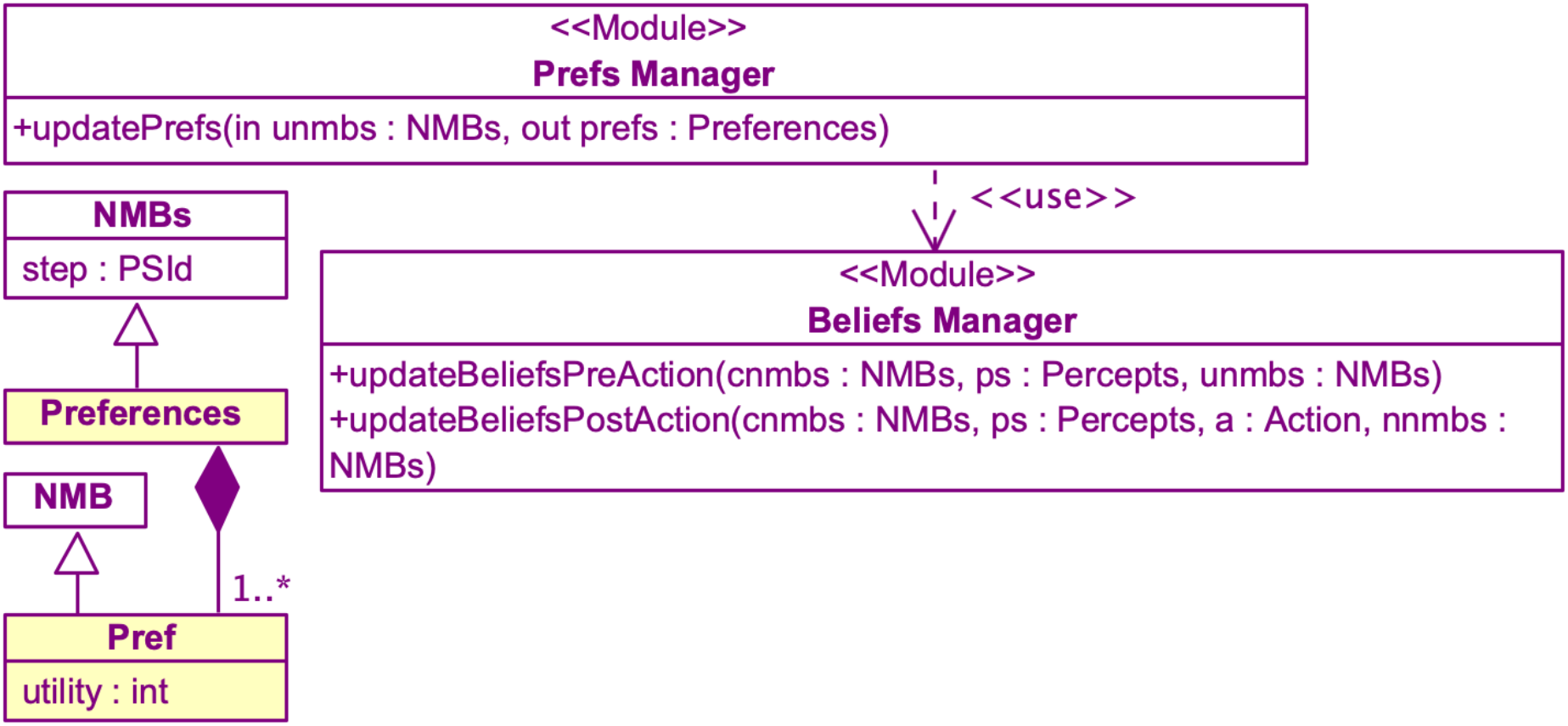
- Updated by the simulator agent
- No uncertainty

WW hunter's beliefs



- Updated (unmbs: Updated NMBs) by the hunter agent before choosing an action (only based on Percepts and cnmbs: Current NMBs)
- NMBs = Non-Monotonic Beliefs
- Completely updated (nmbs: Next NMBs) after the hunter chose an action
- Partially uncertain knowledge

WW hunter's preferences



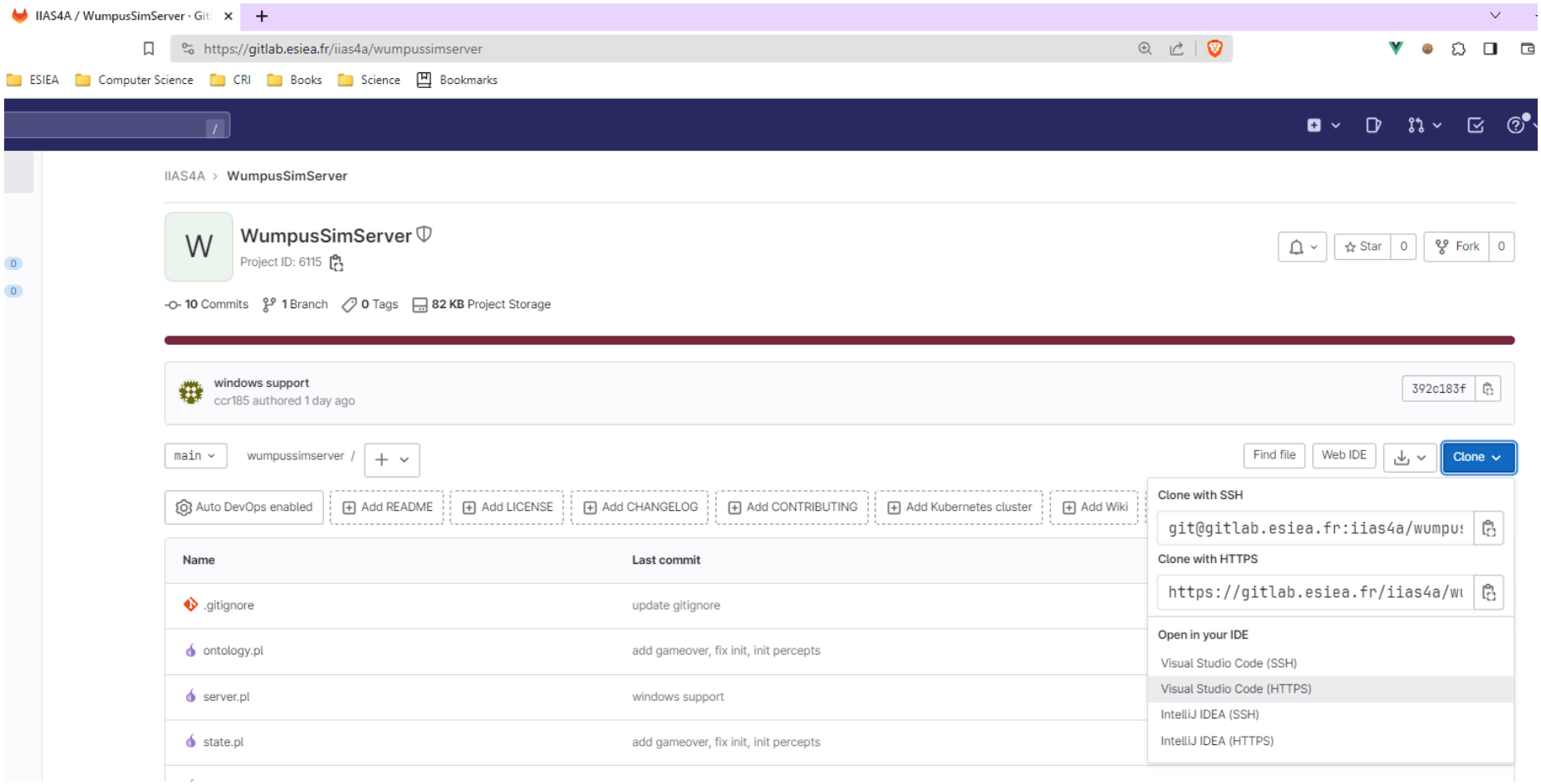
- Adequate preference change
  - From the beginning of the game, prefer WW state in which hunter is alive and is located in the same cell than the gold
  - After the hunter has found and grabbed the gold, prefer WW state in which hunter is located at exit cell(1,1), same as initial position

HW2 Topic

- Develop a WW hunter agent
- Using:
  - The provided WW Simulator agent as an example of web CLP agent
  - The provided WW Web GUI as high-level visual debugging assistant
- The WW Simulator is available on ESIEA's GitLab at:
  - https://gitlab.esiea.fr/iias4a/wumpussim
- The WW GUI is available on ESIEA's Gitlab at:
  - https://gitlab.esiea.fr/iias4a/wumpusgui

HW2: how to use provided code repos

☛Clone WumpusSimServer from Gitlab in another VSCode instance

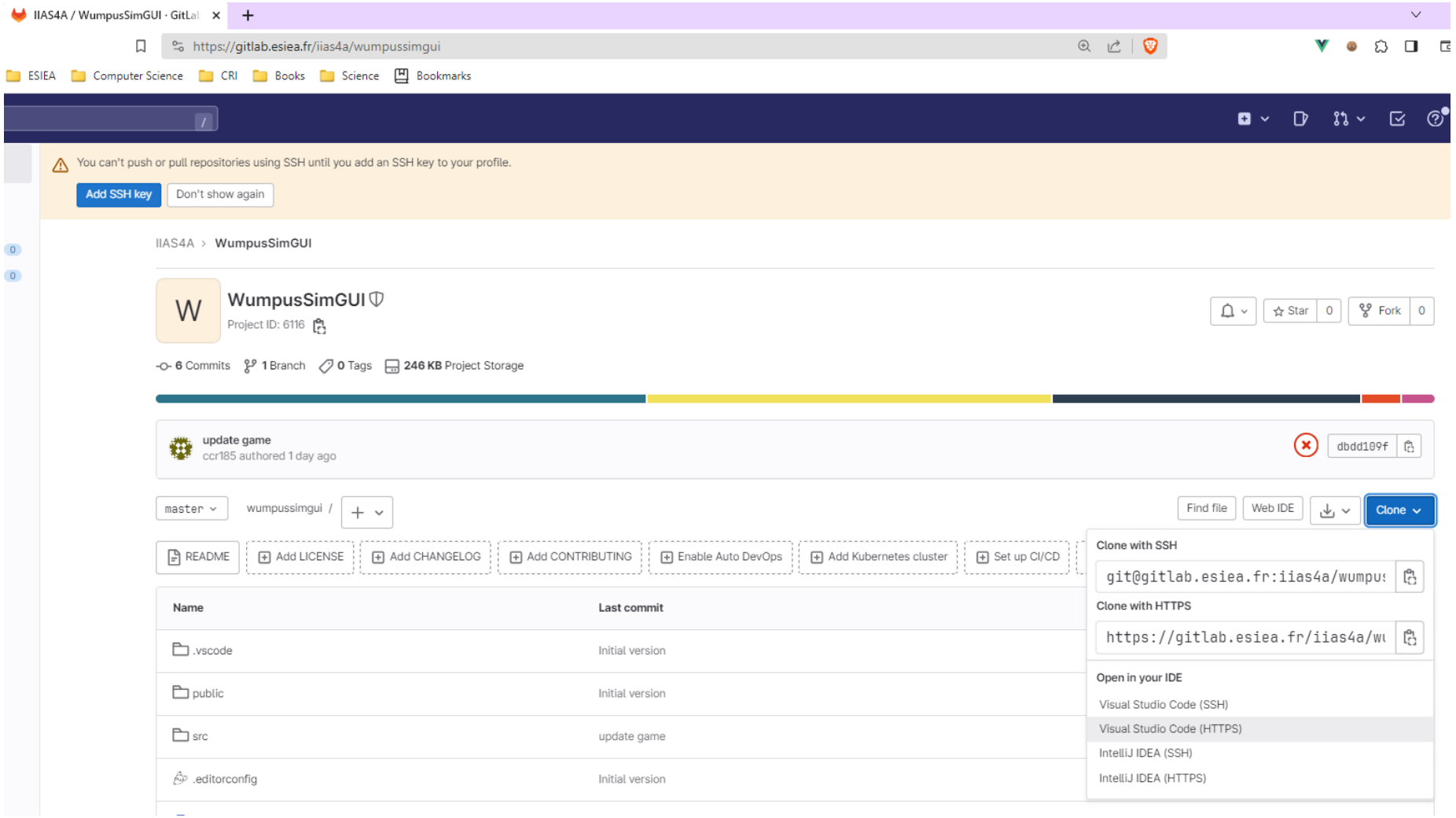


HW2: how to use provided code repos

- Open a VSCode terminal
- Run the command: swipl
- This launches the SWI-Prolog run times which prompts you for queries
- At the query prompt type: [server].
- This will load in SWI-Prolog the code of the WW Simulator Web Service
- Then: run.
- This loads launches the WW Simulator Web Service, waiting from HTTP request from the WW Web GUI.

HW2: how to use provided code repos

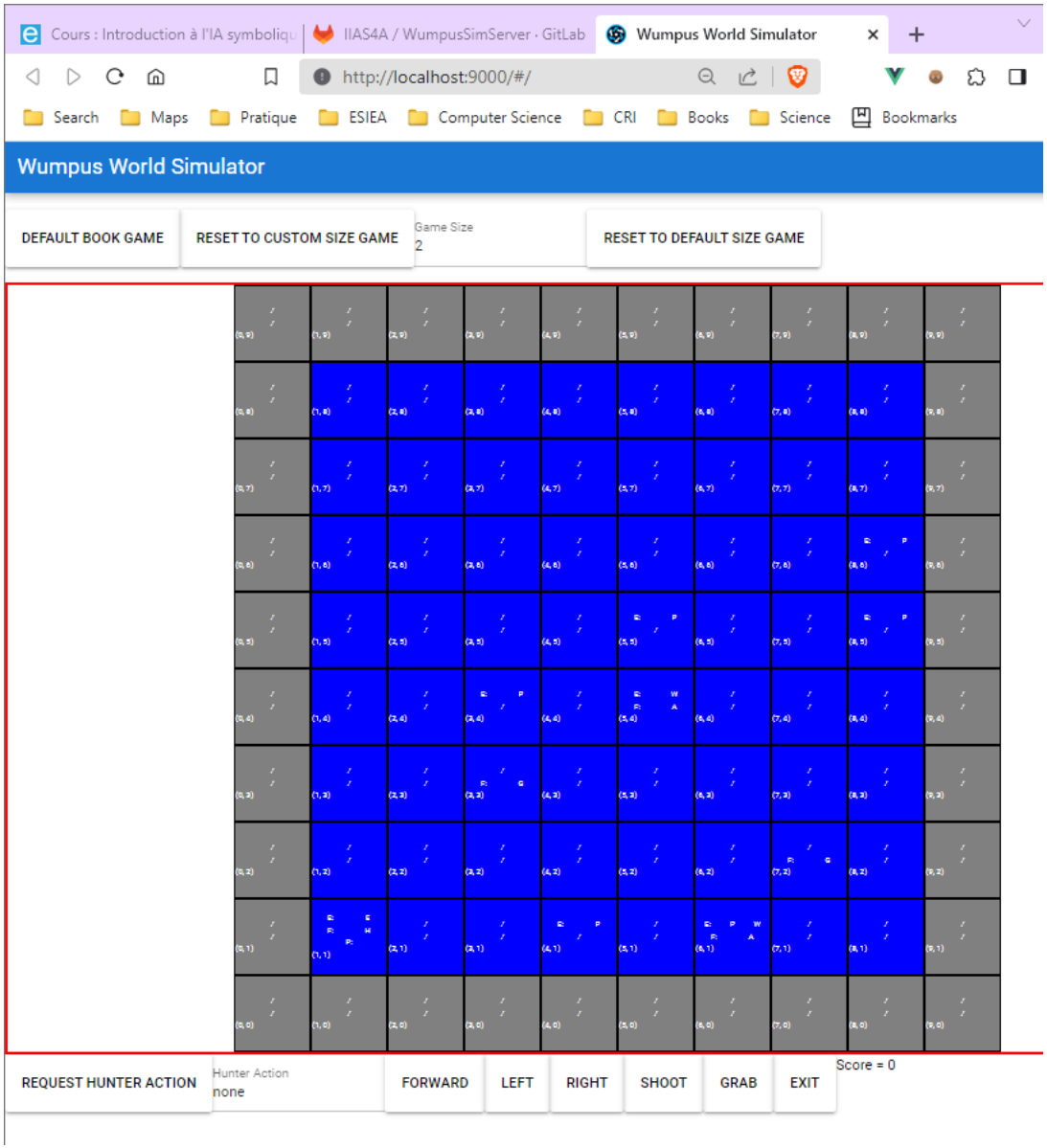
☛Clone WumpusSimGUI from Gitlab in one VSCode instance



HW2: how to use provided code repos

- Open a VSCode terminal
- Install Node.js
- Run the command: npm install
- This will install the Quasar frontend web framework and automatically install all the needed dependencies for the Wumpus Web GUI
- Run the command: quasar dev (or npx quasar dev)
- This will launch the Wumpus Web GUI
- To see it running in a browser, click on link: http://localhost:9000/

HW2: how to use provided code repos



- To initialize the simulation with the 4x4 default world from the AIMA4 book click on Default Book Game button
- To initialize the simulation with an arbitrary WW of size N, click on the button RESET TO CUSTOM SIZE GAME and choose its size with the size counter button located on its left

HW2: what to use and avoid

- Use SWI-Prolog with the libraries:
  - clpfd for declarative constraint-based arithmetic #=. #\=, #>, #<, #>=, #=<
  - dif for declarative non-unifiable
  - pairs for pair processing
  - lists for list processing
  - reif for declarative conditional predicate if_/3 (to add manually to your repository from http://www.complang.tuwien.ac.at/ulrich/Prolog-inedit/swi/reif.pl)
- Do not use:
  - Imperative KB update predicates (assert, retract, etc.),
    instead pass all computation context as predicate arguments
  - Prolog's baseline imperative arithmetic predicates (is, >, <, =:= etc.),
    instead use CLP(FD) constraints prefixed with #
  - Prolog's baseline imperative if-then predicate ->/2 for conditional premises,
    instead use CLP if-then-else predicate if_/3 from reif CLP library
  - Prolog's baseline negated unification \=/2,
    instead use CLP dif/2 of dif CLP library

HW2: warning

- SWI-Prolog dicts cannot be used inside the probabilistic section of a CPLInt program between the
  :- begin(lpad)
  :- end(lpad)
- To reuse your IIAS4A Wumpus World code, you must thus convert the dicts into standard Prolog terms in order to use them in this section of the CPLInt program