Explainable Agents as Static Web Pages: UAV Simulation Example

EXTRAAMAS-2020, New Zealand, 9 - 13 May 2020

Yazan Mualla Timotheus Kampik Igor H. Tchappi Amro Najjar
Stéphane Galland Christophe Nicolle

Université de Bourgogne Franche-Comté - UTBM - CIAD
90010 Belfort cedex, France - www.ciadlab.fr

OUTLINE

- **Future AI Systems**
  - Humanly intelligible
  - Social
  - Trustworthy

- **eXplainable Artificial Intelligence (XAI)**
  - Methods and techniques in AI to facilitate the understandability of AI systems by humans
  - Explainability is one of the cornerstones for building trustworthy and responsible AI systems

- **Explainable autonomous agents**
  - Explain the behavior of agents to humans
  - Increase understandability, transparency, and trust of agents by humans

XAI evaluation

- Approaches in Human Computer Interaction (HCI) studies
- Agent-based Simulation (ABS) as an implementation framework

- **Explainable autonomous agents**
  - Explain the behavior of agents to humans
  - Increase understandability, transparency, and trust of agents by humans

XAI evaluation

- Approaches in Human Computer Interaction (HCI) studies
- Agent-based Simulation (ABS) as an implementation framework

- Lack of interpretability of both black-box machine learning models and complex rule-based systems
- Emerging laws and regulations, *e.g.* European Union's GDPR, require that certain decisions of AI systems must be humanly interpretable
- Scarce HCI contributions that empirically evaluate XAI systems

Goal

Facilitate HCI studies in implementing explainable agents and Multi-Agent Systems (MAS) that can be:

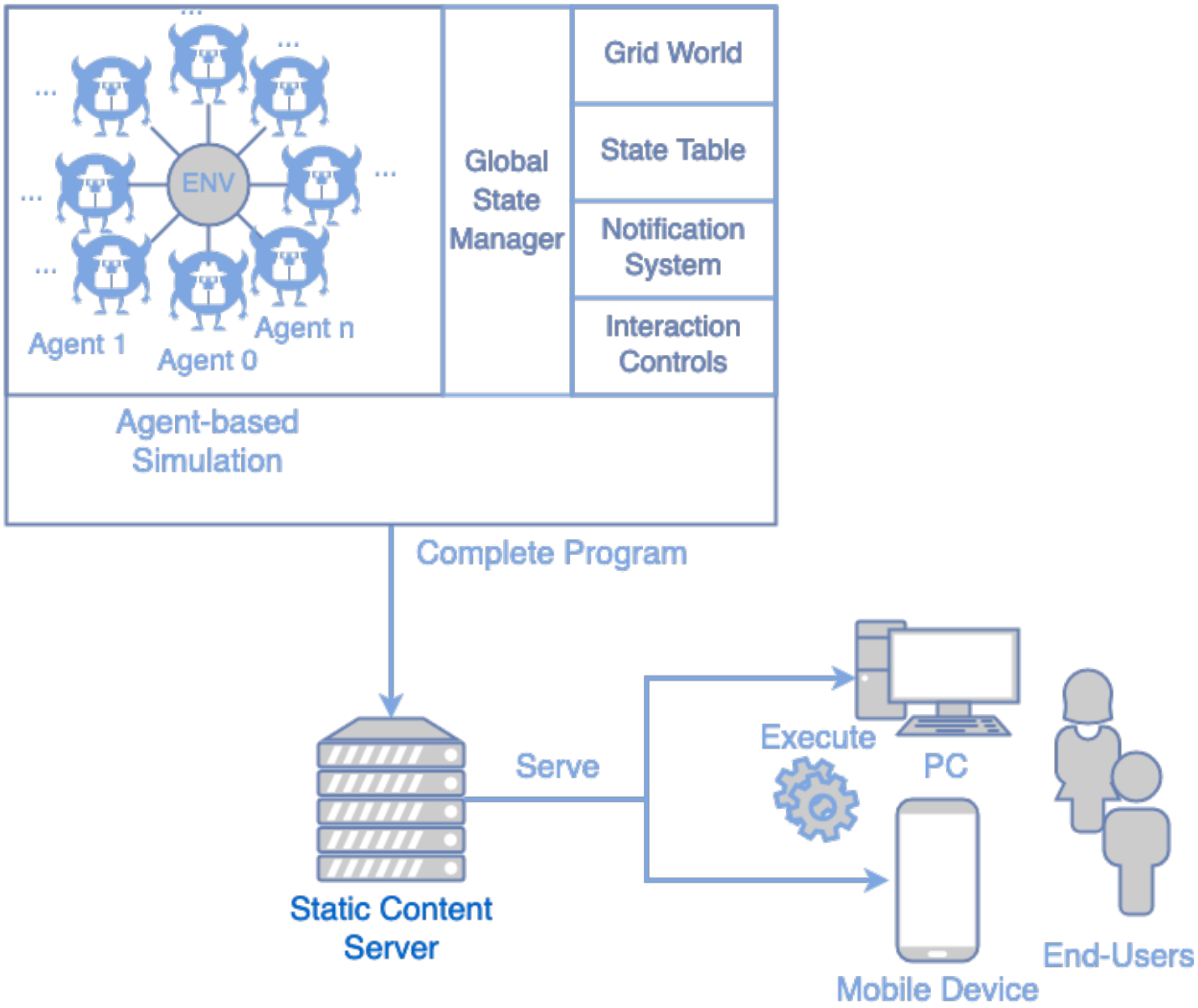
- Deployed as static files
- Embedded into web front ends and other JavaScript-enabled user interfaces

Process

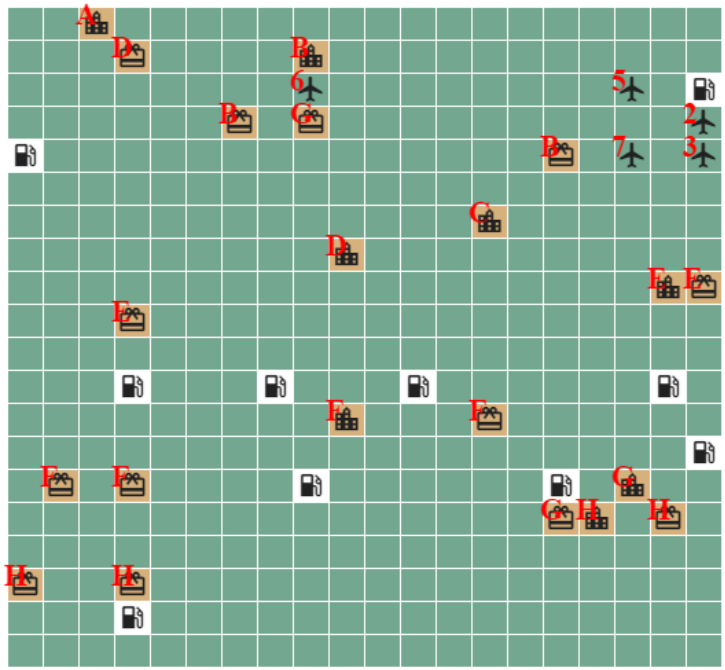
Demonstrate the approach in a simulation of Unmanned Aerial Vehicles (UAVs) designed to assess the effect of different explainability approaches on human intelligibility

- **Ease of deployment:** static files can be deployed in a straightforward manner
- **Reach:** shared with any potential human user who can access the Internet
- **Scalability:** the program code is executed by the client, and the server provides few static files; hence better scalability

- The state of the environment and all agents it contains is exposed to a User Interface (UI) manager component
- UI processes the state and makes it available to the following components:
  - A **grid world** displays the *physical state* of the environment, *i.e.* the position of agents and artifacts
  - A **state table** provides an overview of relevant information that is not obvious from the grid world representation
  - A **notification system** informs the users about important events. Notifications are displayed as visually invasive alerts
  - **Interaction controls** allow users to switch between different simulation modes and adjust simulation parameters


ENV
Agent 1
Agent 0
Agent n
Global State Manager
Grid World
State Table
Notification System
Interaction Controls
Agent-based Simulation
Complete Program
Serve
Static Content Server
Execute
PC
Mobile Device
End-Users

- The simulation makes use of the JS-son library [Kampik and Nieves (2019)]
- The simulation is provided in three modes:
  - **Basic mode:** It displays the current state of all agentsin a table-like overview that updates in real-time
  - **Adaptive filter mode:** It aggregates the most important information across agents. When an agent enters a possibly problematic state, an alert is generated
  - **Contrastive mode** Alerts are constructed using a contrastive explanation scheme, following the structure:
    $Agent\ A\ is\ doing\ P\ [instead\ of\ Q]\ because\ of\ C$
    where P is the current behavior, Q is the expected behavior, and C is the execution condition



Drone Delivery Simulation

⚠ **No clear target** for drone(s) **7**: assigned package(s) just picked up by other drone(s).

1 drone(s) on way to package :	1 drone(s) on way to target :	5 drone(s) need(s) re-charge :	2 drone(s) idle :
3	2	0,1,4,5,6	8,9

Legend:

- Drone
- Package to be picked up
- Delivery destination
- Charge station

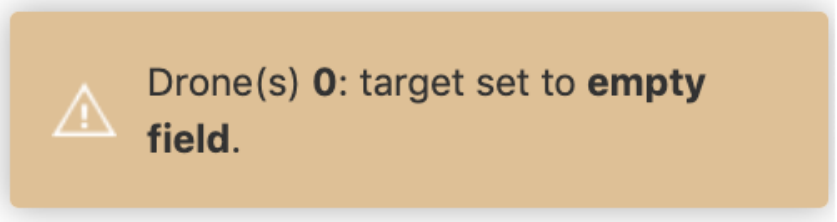
Controls (click "Restart" to update):

#Drones: 10 drones | Choose a mode: Adaptive filte | Speed: 2 | Seed: 5

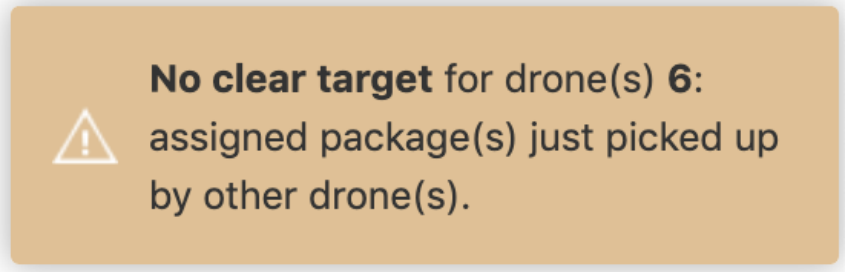
RESTART | GET LOG (RIGHT CLICK)

Drone No.	0	1	2	3	4	5	6	7	8	9
Battery	36	36	36	38	40	36	36	36	36	36
Package loaded?	No	Yes	No	No	No	No	No	No	No	Yes
Current task	Go to 13	Go to 2	Go to 59	Go to 59	Go to 80	Go to 29	Go to 123	Go to 99	Go to 177	Go to 178
Task type	Go to plain	Go to target	Go to station	Go to station	Go to station	Go to plain	Go to package	Go to package	Go to package	Go to target
Position	12	8	55	49	48	26	124	85	174	166
Location type	plain	plain	plain	plain	plain	plain	plain	plain	plain	plain

1 drone(s) on way to package :	4 drone(s) on way to target :	5 drone(s) need(s) re-charge :	0 drone(s) idle
7	1,6,8,9	0,2,3,4,5	



Basic alert to explain unexpected behavior



Contrastive alert to explain unexpected behavior

- Demo link: `http://s.cs.umu.se/51x65y`

OUTLINE

- Explainable agent simulations can be deployed as static web pages
- Light-weight tools with a small development, deployment, and operations footprint can be utilized to:
  - Rapidly develop explainable agent prototypes in a widely-used higher-level programming language
  - Roll-out these prototypes and simulations at scale to large and diverse user groups

- From an engineering perspective, extend the JS-son library with additional, generically useful abstractions for implementing explainable reasoning-loop agents
- From HCI and XAI perspectives, extend the simulation to allow for human-in-the-loop feedback

Thank you for your attention...

Appendix

Bibliography (#1)

Kampik, T. and Nieves, J. C. (2019). Js-son-a minimal javascript bdi agent library. In *7th International Workshop on Engineering Multi-Agent Systems (EMAS 2019)*.