Policy-Making Environment

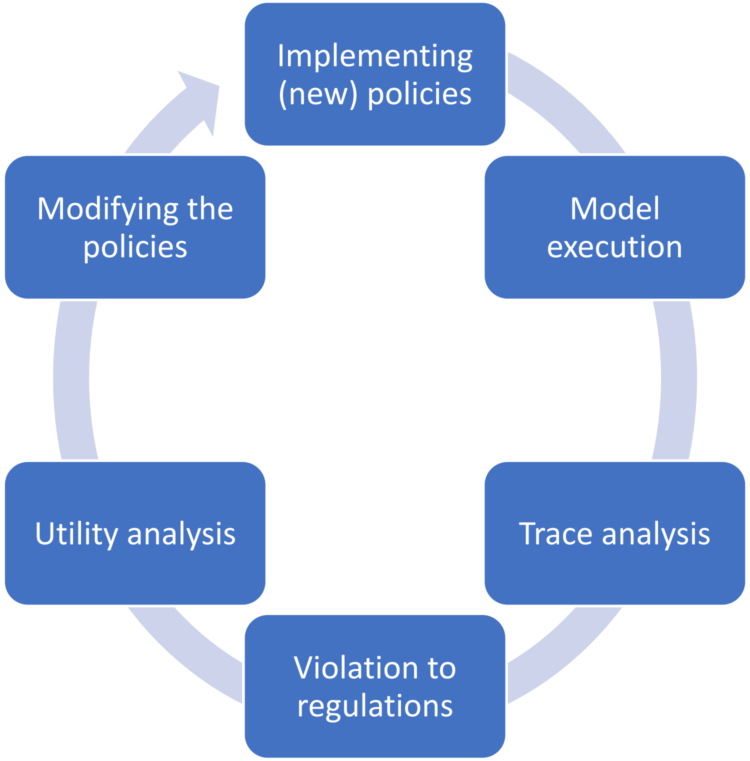
Mostafa Mohajeri

University of Amsterdam

June 28th, 2021

My position in DL4LD

- Design and develop a computational environment for **policy-making**
- To test the effects of regulations and policies
- Focus: Utilizing agent-based models of the social actors


Implementing (new) policies

Model execution

Trace analysis

Violation to regulations

Utility analysis

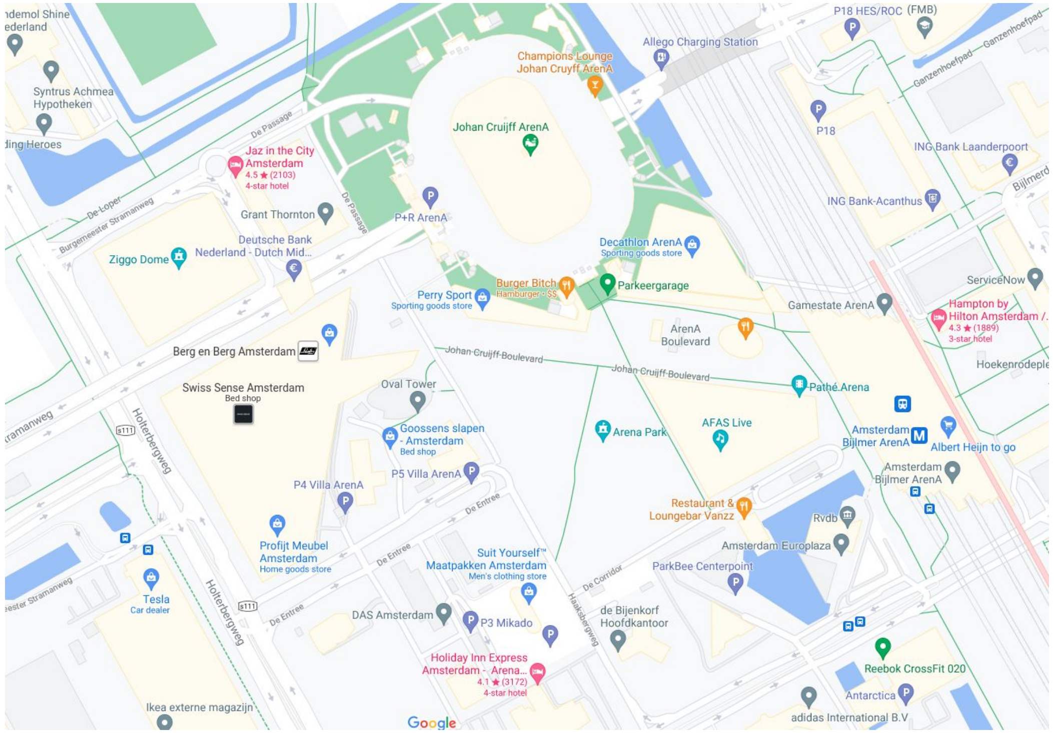
Modifying the policies

Overview

- Agent-based programming framework as part of the policy-making environment
- Aim of the framework:
    - Using MAS theory to model a social setting
        - Explainable autonomous agent
    - Verification of policies via model execution
- Practical requirements:
    - Scalable for large models
    - Productive for System/Policy designer
- Developing use cases as proof-of-concept
    - KYC case (SSPDDP)
    - Amsterdam Arena Mobility Case

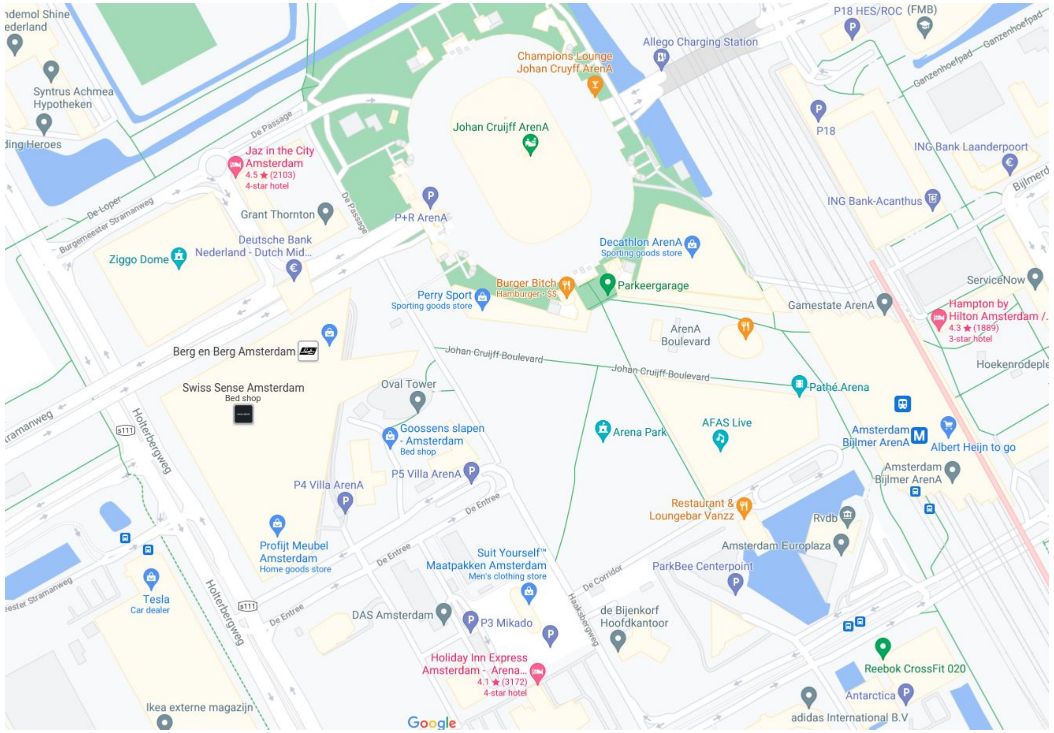
The example use-case: Arena

- Example agents of the social system:
  - Operational Mobility Center (OMC)
  - Drivers
  - Public Transport
  - Police
  - Parkings
- Interact and Communicate
  - With other agents
  - With the environment



The example use-case: Arena

- Example Behavioral Norms and Policies of the system
  - Policies are being designed/tested
    - Data sharing policies
      - What data can be shared with whom
      - What data is needed to be shared
      - For what purpose can data be used
    - Traffic routing policies
  - Assumptions of the model
    - Data sharing regulations (GDPR)
    - Data controllers’ policies
      - Telecom companies, Public transport
    - Driver’s behaviors



Research Challenges

- How to create an *executable* and *explainable* models of agents?
- How to interface the agents with the institutional reality?
  - Reasoning about policies and regulations
- How to model the utilities of the agents?
  - Their preferred actions or state of the world
- How to make the model execution scalable?
  - Explainability often conflicts with scalability
- How to make the policy design/test cycle efficient?
  - Making the framework *usable*

Challenge 1: Executable model of Agents

- AgentScript Cross-Compiler framework (ASC2)
- A multi-agent system (MAS) framework to model the actors
    - A logic-based programming language
        - Readable and Verifiable
    - Intentional agents based on Belief-Desire-Intention (BDI) model
        - Transparency w.r.t. intention behind their actions
        - Feeds back to the policy-making

ASC
AgentScript
Cross-Compiler

Challenge 2: Normative Agents

- Agents should reason about Norms
  - Regulations, Contracts, Policies
  - Reason about their permissions, powers and duties
  - Act based on them
  - Know when they are violating them
  - Know when another agent is violating them
- ASC2 is Interfaced with ***norm reasoning*** frameworks like ***eflint***
  - We have already done a few example cases in the SSPDDP project

Challenge 3: Preferences

- Agents should act based on **explicit preferences**
  - Required for conflict resolution
  - Specially when policies are in conflict with each other
    - Cars should be distributed between parkings evenly
    - Cars should be routed in a way that creates the least traffic
- Explicit preferences make the decisions making transparent
  - Published paper: **Integrating CP-Nets in Reactive BDI Agents, (**PRIMA2019)
  - Published paper: **Declarative Preferences in Reactive BDI Agents, (**PRIMA2020)

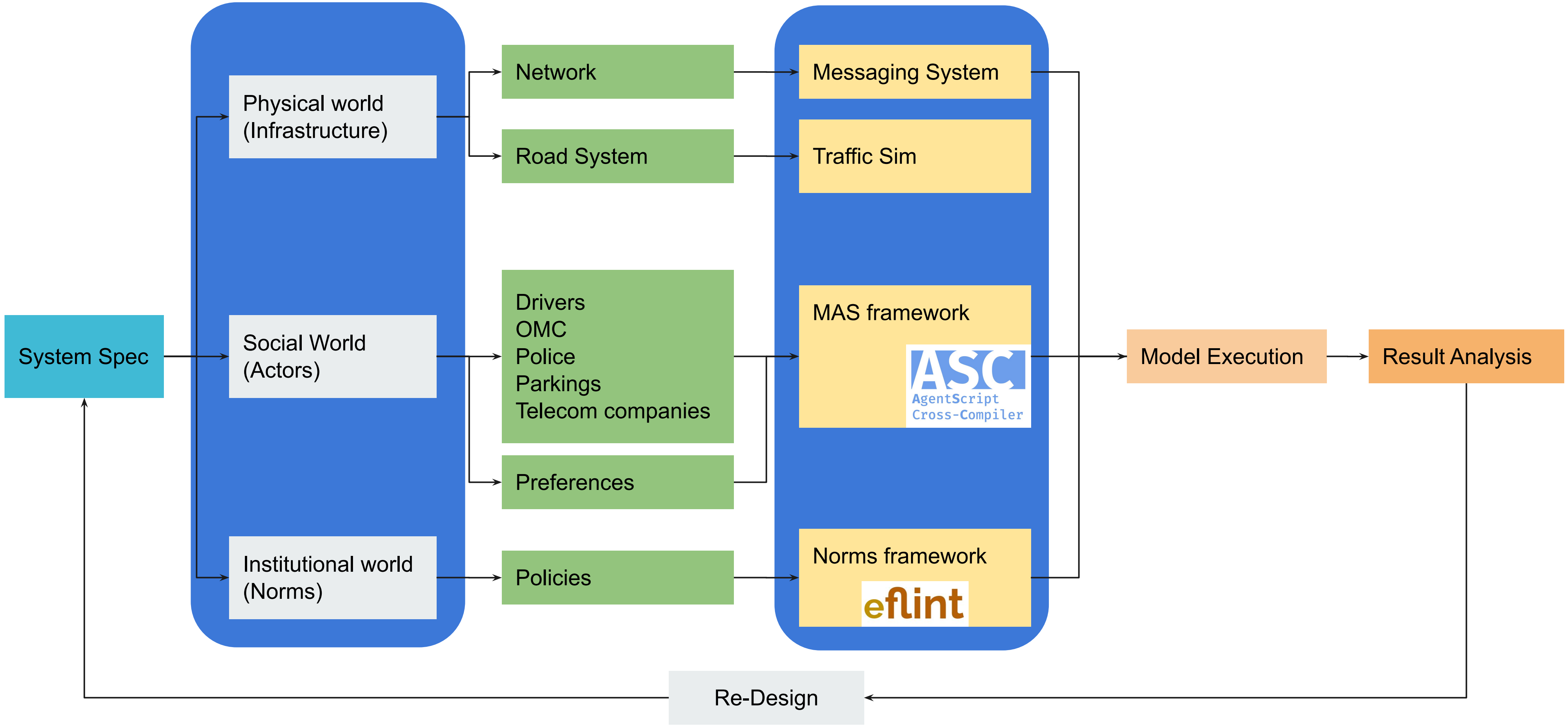
Challenge 4: Scalability

- Explainability often has conflict with scalability
  - Current agent-based programming frameworks are not scalable enough for larger models
- ASC2 acts as a compiler
- Translates the high-level language to executable programs
  - High-level language promotes readability and simplicity
  - Low-level language guarantees performance
    - Able to execute in a distributed setting
- Published paper: **Run Agent, Run! Architecture and Benchmarking of Actor-Based Agents** (AGERE@ETAPS2020)

Challenge 5: Productivity

- Testing and integration of the System model and policies can be cumbersome
    - Norm framework, ASC2, other 3rd party software and services
    - The focus should be on the design itself
- ASC2 can utilize online DevOps systems, e.g. TravisCI, CircleCI
    - To build-up a system from multiple repositories
    - Integrate with other system, e.g, norm framework, traffic simulators
    - Automatically Run tests and record the results
    - Keep the logs for future reference
- Published paper: **Seamless integration and testing in MAS Engineering** (EMAS@AAMAS2021)

Example Design Cycle



Thank you :)