Policy-Driven System Design

Mostafa Mohajeri
University of Amsterdam

CCI Meeting
Feb, 2022
Project: LICCAM

- Legal Intervention in Connected Cooperative Automated Mobility
- Creating a demo implementation of the system
  - Monitoring roads and autonomous vehicles
  - Able to reason about possible future high risk states
  - Able to reason about possibility of intervention
    - Through 3rd party controllers (OEMs)
  - The legal process is part of the technical process

- My goals:
  - Exercising policy design as part of system design
  - Focus: Utilizing agent-based models of actors to reason about policies
Target System Attributes

- **Socio-Technical**
  - Software Actors: service providers, routers, sensors
  - Social Actor: organizations

- **Regulated**
  - Regulated Entities: *Actors, Groups of Actors and the System (as whole)*
  - Regulator Entities: Monitors, Enforcers, Auditors

- **Data-Intensive**
  - All communications and their purpose are important

- **Open**
  - Existence of external world affecting and getting affected
Regulated Data-intensive Socio-Technical Open System

Dynamic policy changes affect the behavior of the regulator actors which propagates to the system behavior.

Regulations/Policies dictate the behavior of regulator actors which changes the behavior of the system.
Example Design Cycle

System Spec

Physical world (Infrastructure) → Network, Sensors → Communication Protocol

Social World (Actors) → Controllers (OEMs) Car Owners Gov agencies (RDW)

Institutional world (Norms) → Traffic Regulations, System policies → Norms framework

MAS framework

Model Execution via Scenarios → Result Analysis

System Design Artifacts

Policy Design Artifacts

Re-Design
An Example Case: LICCAM

- Design: Applying the mentioned method in System/Policy design cycle
- Desired output: An executable model of the system containing:
  - Design artifacts
  - Policy artifacts
- The rest of the presentation is a recap of the experience
Initial System Spec
Initial ASC2 Model

We can execute scenarios to verify the system
Decoupling the Environment

We had very limited and predictable scenarios

ASC2 is protocol agnostic

Traffic Simulation

Execute less predictable scenarios
Policies vs. Control

The system as a whole should be verifiable against regulations by using execution traces.

The verification happens on the model at design time where it is still feasible.

Policy and System design feedback to each other.

Example: The OEM should execute an intervention within a timeframe if there is a warrant from oracle.

Policy Design Artifacts

- straightforward
- non-functional requirement

- not so straightforward
- regulation

What is the incentive?
What are the punishments?
What is the evidence?

System Design Artifacts

Policies become part of the system design.
Explicit (dynamic) Policies

Example 1: In normal situations, a warrant for intervention should be issued only with intention to stop a **HIGH RISK** state

Example 2: In extreme situations (terrorist attack), a warrant can be issued in any intention

Some actors act based on explicit norms, specially actors with dynamic policies

They change the system behaviour by changing policies

To have a formal specification of policies
Even monitoring the environment can create liability! even more in affecting it
Do we want this system?
Usability: Automated Tests

```
"An oracle" should {
  "always issue a warrant if there are proper evidence" in {
    val oem = agents("oem1")
    // Send the request to oracle
    agents("oracle").send(warrant_request("case_1","set_speed(90)","oem1"))
    // Intercept response
    val message = oem.receiveMessage()
    // Assert
    assert(message instanceof GoalMessage)
    assert(message.toString() startsWith "warrant(execute(set_speed(90),car1")
  }
}
```
Conclusion

- Applied an ABM approach to System/Policy design cycle
- Policy and System design should be done together
  - They are affected by each other
  - They feedback to each other
    - e.g., need for evidence requires adding monitoring
- Just like software tests, compliance verification can not be an afterthought
  - More challenging to test and verify
  - Much more challenging to fix
Policy-Driven System Design

Mostafa Mohajeri
University of Amsterdam

CCI Meeting
Feb, 2022
Thank You! :)