# Towards Regulatory Design with Agent-Based Modeling at the Boundary of Law and Software

1st Workshop on Agent-based Modelling and Policy-Making (AMPM), in conjunction with JURIX 2021

Sebastian Benthall, New York University School of Law
Joshua M. Epstein, New York University School of Global Public Health
Erez Hatna, New York University School of Global Public Health
Katherine J. Strandburg, New York University School of Law
Michael Carl Tschantz, International Computer Science Institute

Supported by NSF # 131532, Designing Accountable Software Systems

# ABMs for software accountability

We propose using agent-based models (ABMs) to improve software accountability.

Currently, software accountability measures lack a systematic way of predicting societal impact.

ABMs are a way to model the societal impact of software.

**Potential:** ABMs are legible to software engineers, social scientists, and regulators.

**Challenges:**

- Choosing the ABM under political pressure.
- Measuring robustness of impact.
- Designing the interface between software and ABM.

# Software accountability

*Software accountability* refers to the problem of holding software systems accountable to laws and regulations.

More and more individually and socially significant private sector activities are automated, e.g.:

- Employment
- Lending
- Targeted advertising
- Higher education

These areas are subject to sectoral law.

*Omnibus* data protection laws such as the E.U. General Data Protection Regulation (GDPR) and California Consumer Privacy Act (CCPA) are unsettled.

Many open, interrelated problems of: regulatory design, software design, and compliance testing.

# Software accountability is difficult (1)

Software challenges to traditional approaches to accountability.

Software and law are written in literally different languages.

What have people tried? Increasing transparency of:

- when and how software is used
- data provenance and feature verification
- relationships between inputs and outputs
- software code itself

But it is not easy to understand what these aspects of the software system mean, with respect to regulations.

Some regulatory compliance *cannot* be done only by looking at the software.

- References to beliefs, meanings, intentions
- Regulatory goals aim at social consequences, not deontological rules

# Software accountability is difficult (2)

Other approaches:

- “algorithmic impact assessments”
- summaries modeled on nutrition labels
- black box testing

There is no systematic mechanism for exploring the complex interactions between the software, other aspects of the regulatory system and the affected social systems.

What about testing software with multi-agent systems (MAS)?

# The importance of the social environment

There are three elements to the design of accountable software systems:

- the software system itself;
- the regulatory environment, including regulations, regulators, and their enforcement mechanisms.
- the social environment in which it operates;

The appropriateness of the software depends upon the software's impacts on people in the social environment.

Accountable software regulation thus requires expertise relating to all three elements:

- software engineers
- regulators
- domain social scientists

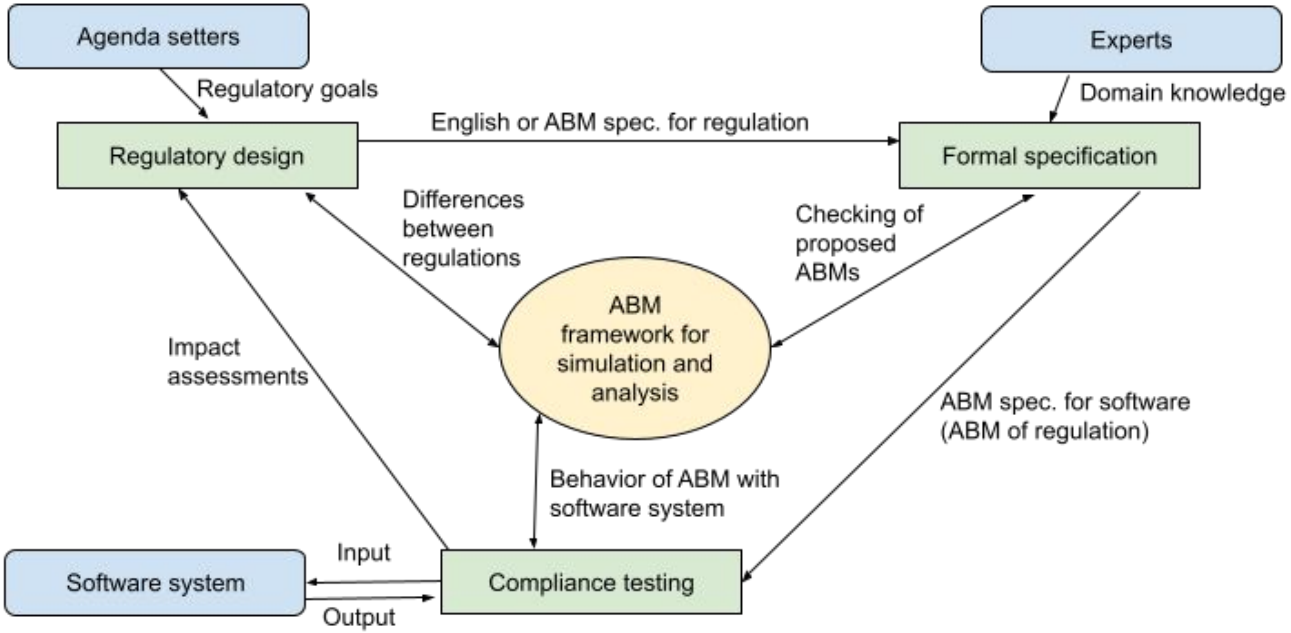
# Why ABMs for software accountability

ABM is already part of the policy-making toolkit.

ABMs may have additional advantages when *software* is a regulated object.

- ABMs can model the social environment, including people's beliefs and purposes. These are often the referents of law, and not elements of the regulated software per se.
- ABMs can illustrate the social impact of a software system in a way that is more intuitive to regulators and domain experts than software performance metrics.
- ABMs can replicably account for social complexity and feedback loops.
- ABMs are implemented as software artifacts, and so are intelligible to software engineers and potentially interoperable with regulated software.

ABMs can facilitate dialogue between engineers and regulators about the goals and meanings of regulations.




Workflow with our three proposed components (regulatory design, formal specification, and compliance testing) as stages of a general method for enforcing software accountability. This figure illustrates one potential configuration of these components with respect to the regulation of a private sector software system. (Scope of NSF # 131532)

# A process

(1) A regulator that has been entrusted with social goals in a particular domain.

(2) The regulator works with domain scientists, who provide a descriptive ABM that include actors and social outcomes.

(3) The regulator introduces normative elements into the model, which may be:

- hard rules on agent behavior or
- objective functions to be tracked and optimized.

This model can be used to analyze the intended and unexpected consequences of the regulation.

# The task ahead: regulatory design

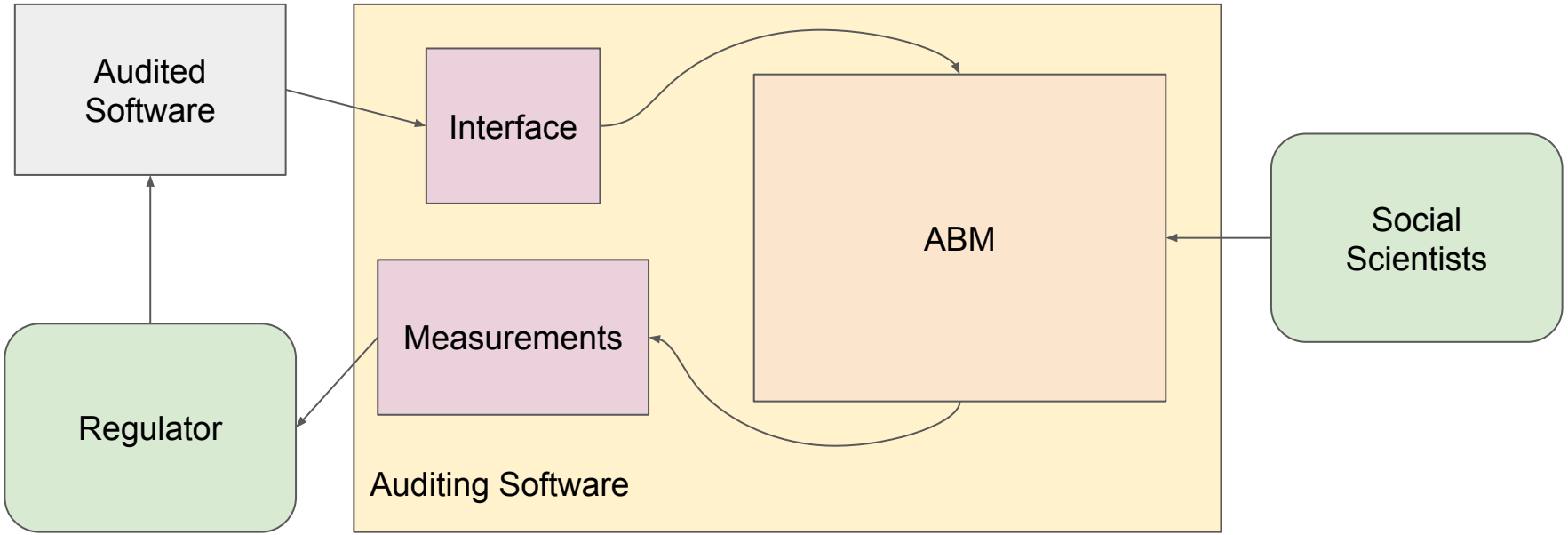
The goal: a *regulatory design process* that involves ABMs, spanning the law, software, domain expertise.

Focus on the potential uses of ABM in a legal context of reporting, auditing, and enforcement of regulation aimed at software systems.

Points of contact between software and regulation that are unsettled law:

- *Ex ante* design of software systems
- Regular “impact assessment” reporting of social impact of software systems
- Response to investigations

We explore the legal and technical feasibility of software that allows regulatory users to interact with and visualization results from the model.



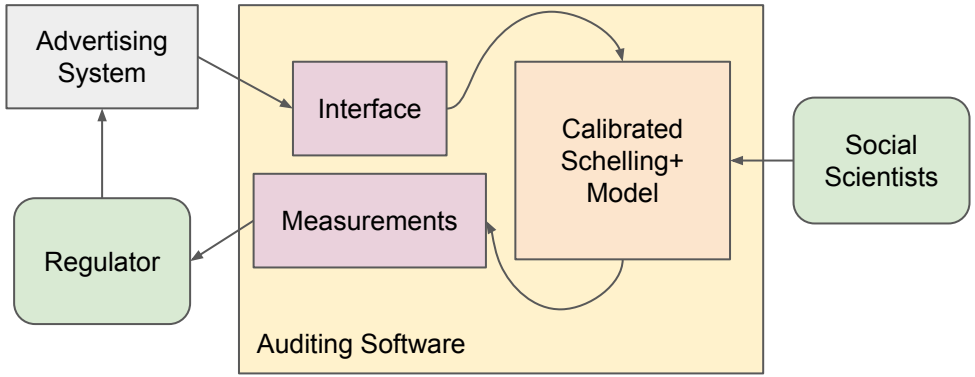

Proposed architecture for software auditing system.

# Example: Housing Advertising

Housing advertising is regulated in the U.S. by the Fair Housing Act. It prohibits discrimination in advertising. Many of the applications of this rule are unsettled, especially with respect to digital advertising.

One goal of this regulation is to reduce *housing segregation*, which emerges from the social system in combination with advertising.

While black-box testing can show bias in the presentation of advertisements, a calibrated ABM of housing choice can show which dimensions of discrimination (e.g. based on location preference, income, or explicit ethnic discrimination) are most influential on segregation.



# Identifying the challenges

We have identified several key challenges to the proposed approach. These “pain points” are each opportunities for future technical and policy research.

- Choosing the model. The regulated entity? Select experts?
  - Designing and calibrating the model will be politicized.

- Testing the robustness of the software’s compliance to social changes. A methods problem.
  - Quantifying endogenous change within an ABM’s results …
  - … as well as variations over its parameter and rules space.

- Defining the interface between the ABM and the audited software system
  - What inputs/outputs from the software system are used by the ABM?
  - What statistically summaries of software behavior are needed?
  - Can this be done efficiently?

Thank you!