Universiteit van Amsterdam
Instituut voor Informatica

DMI-UvA Research Update

L. Thomas van Binsbergen
l.t.vanbinsbergen@uva.nl

Central Research Questions

"In what ways can we integrate legal/normative decision processes in software systems such that decisions are accountable and scalable?"

Concretely, we *study* and *design*

software **languages**, **architectures**, **protocols** and **algorithms** that

incorporate terms such as:

permission, duty, power, delegation, dispute, evidence, interpretation, …

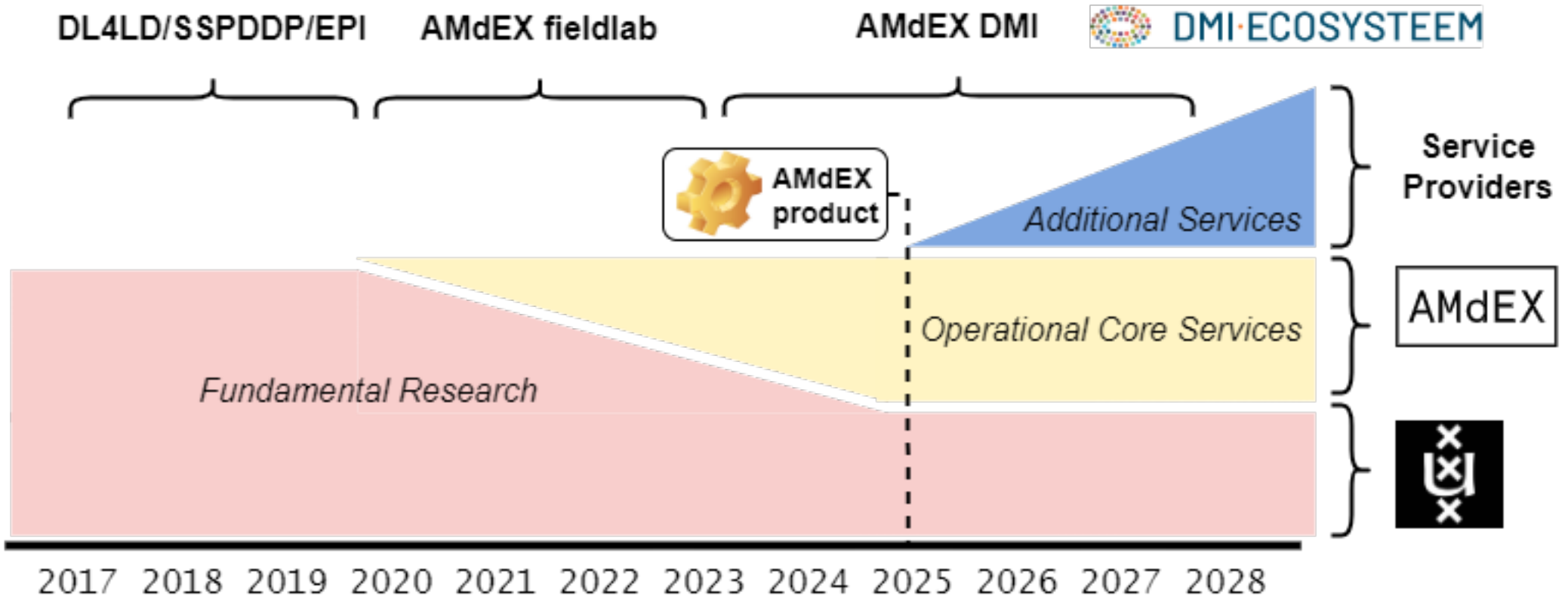
Research Areas

1. Language Design and Norm Engineering
   - Machine-executable representation of laws, regulations, contracts, etc.
   - Core contribution: eFLINT **normative specification** language (and friends)
2. Normative reasoning in Multi-Agent Systems
   - How can we incentivise (human and software) agents to behave compliantly?
     Central insight: positive (carrot) and negative (stick) enforcement of **duties**
   - How can we distribute normative reasoning across agents in a system,
     such that agents agree on whether agents behave in a compliant manner?
3. Software Engineering for Data Exchange Systems
   - Roles, architecture, and protocols for automating compliance in data ecosystems
   - Main deliverables: **AMdEX architecture** and AMdEX governance in **DMI PDX**

AMdEX Reference Architecture

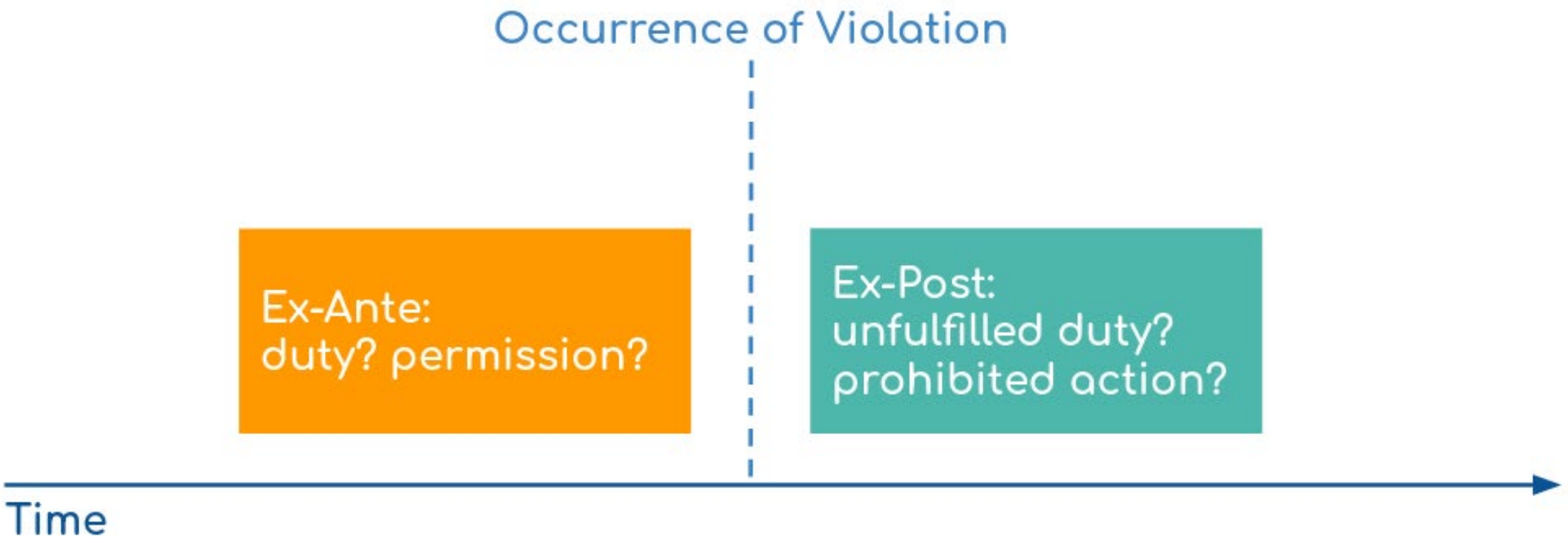
DMI Ecosysteem PDX demo

AMdEX Vision and Timeline

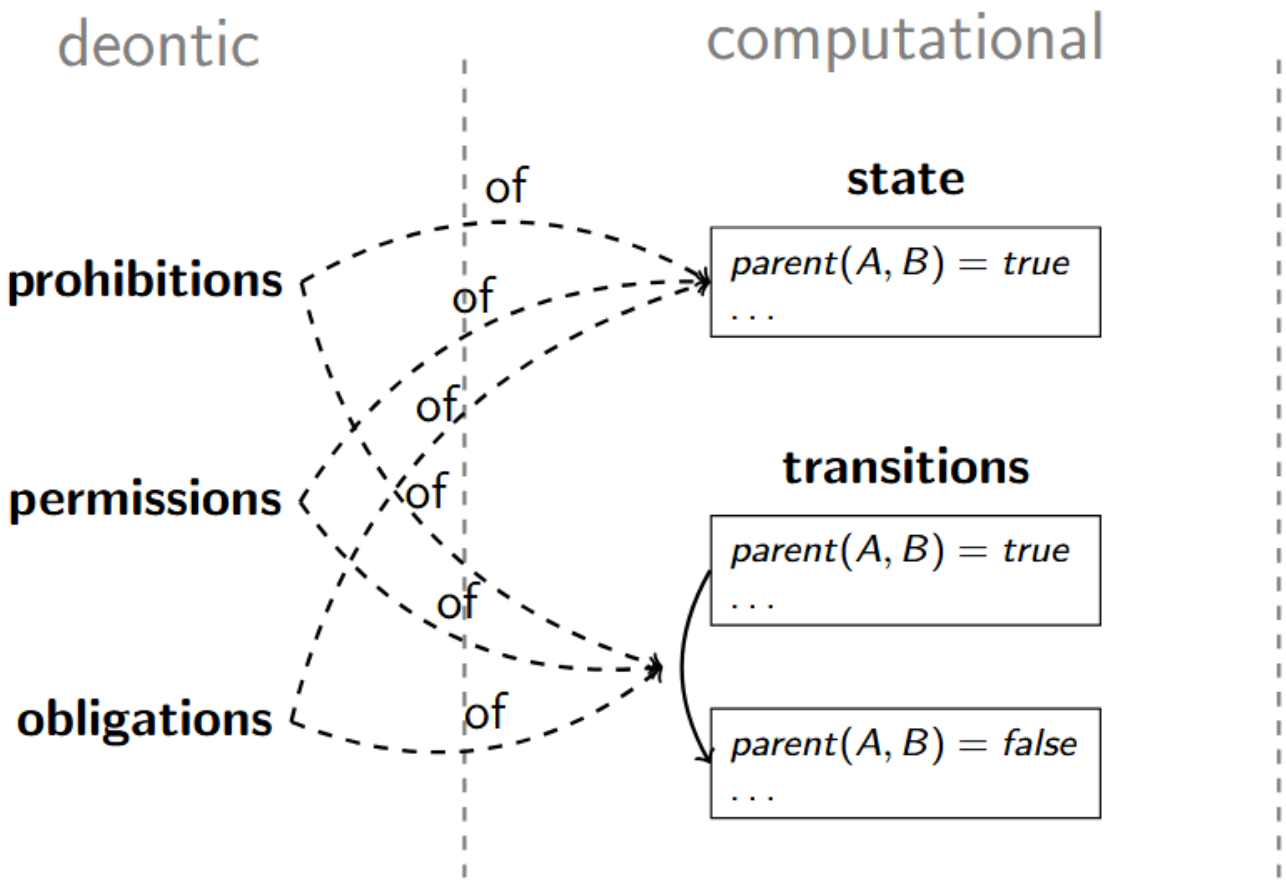


AMdEX product as part of DMI Producten en Diensten eXchange (PDX)

Ex-ante vs Ex-post Enforcement



Linking normative and computational concepts



- Powers have (normative) consequences
- Deontic and potestative terms are first-class
- Powers modify truth-assignments to variables
- Duties capture expected actions

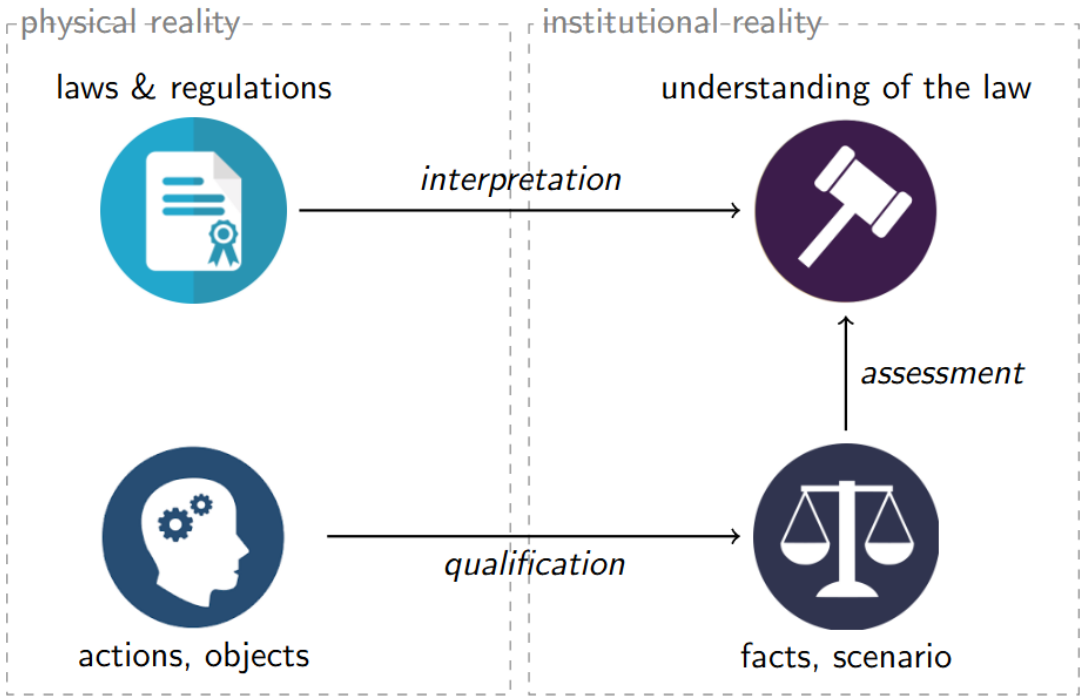
- Violations of state and transition

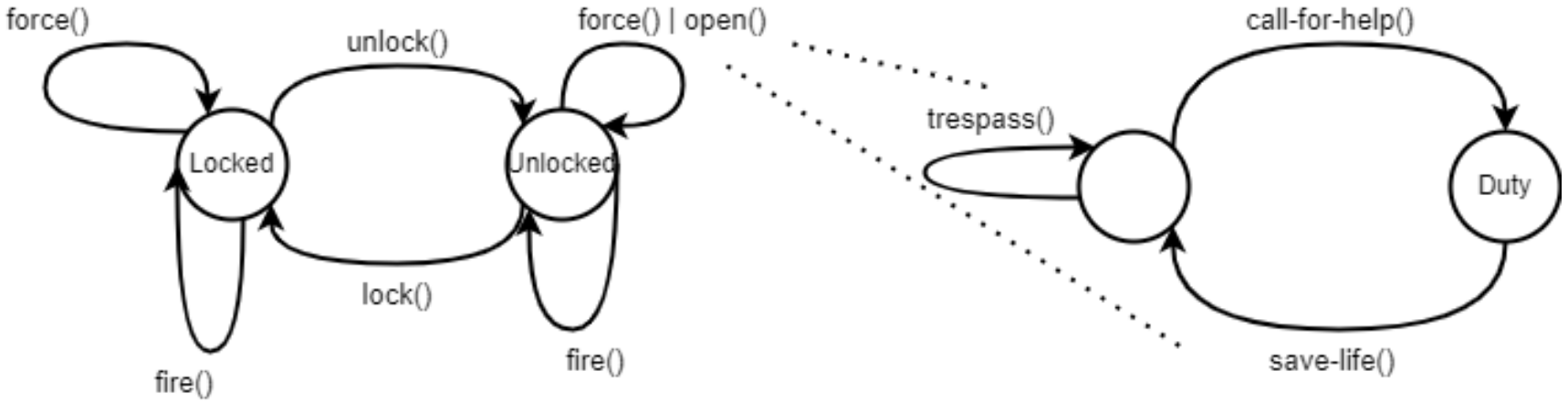
Machine-readable vs Machine-executable

Assessing individual **scenarios**

Verifying system-wide **properties**

Simulation and model-driven eng.
→ compliance by design





"A fireman is trespassing
when forcing a door
if there is no
fire or call for help"

https://doi.org/10.1145/3410256

Norm engineering for data spaces

Norm engineering pipeline:

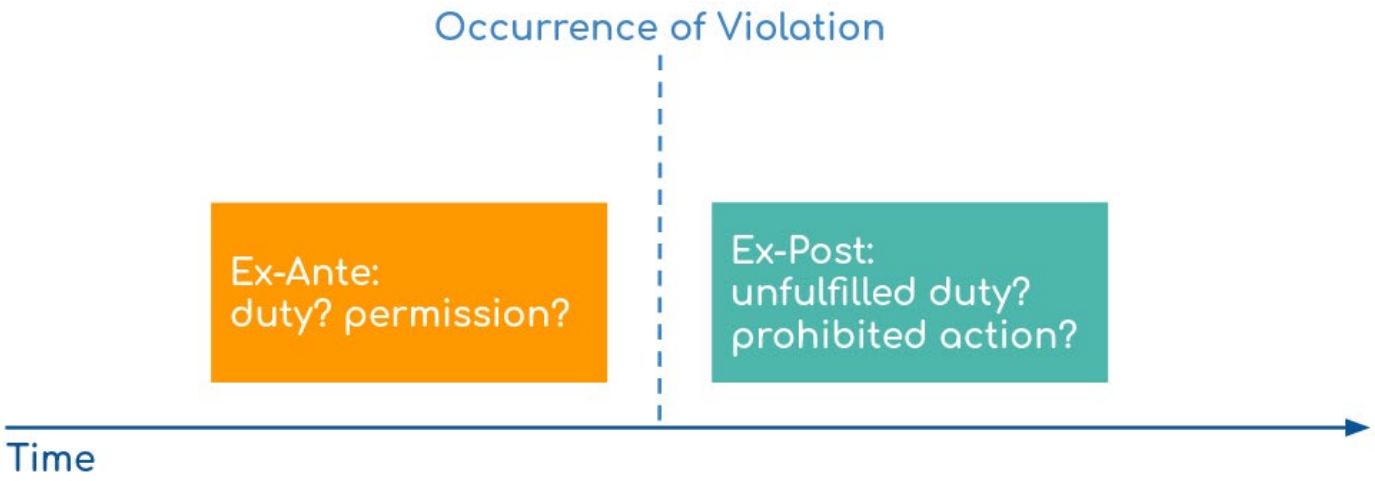
law, regulation, contract, usage condition → formal interpretation (software code)

Digital enforceable contract pipeline:

template(contract x formal interpr.) → legal and executable contract

Integration in AMdEX governance flow (AMdEX Reference Architecture v1)

Enforcement strategies



Enforcement strategies:

- **Static ex-ante**;
  Simulation, orchestration, planning

- **Dynamic ex-ante**;
  Access control

- **Dynamic ex-post**;
  Usage control, runtime verification, adaptations

- **Static ex-post**;
  Accountability and Auditing

Programme

15:00-15:10 uur
Thomas van Binsbergen – Introduction

15:10-15:30 uur
Tim Müller – Intro to the eFLINT policy language and DMI-Commons as executable policy

15:30-15:50 uur
Merrick Oost-Rosengren – Policy registration and information dossiers

16:00-16:20 uur
Christopher Esterhuyse – Policy specification and communication in distributed systems

16.20-16:40 uur
Heleen Janssen – Business-to-Government data sharing via an Intermediary

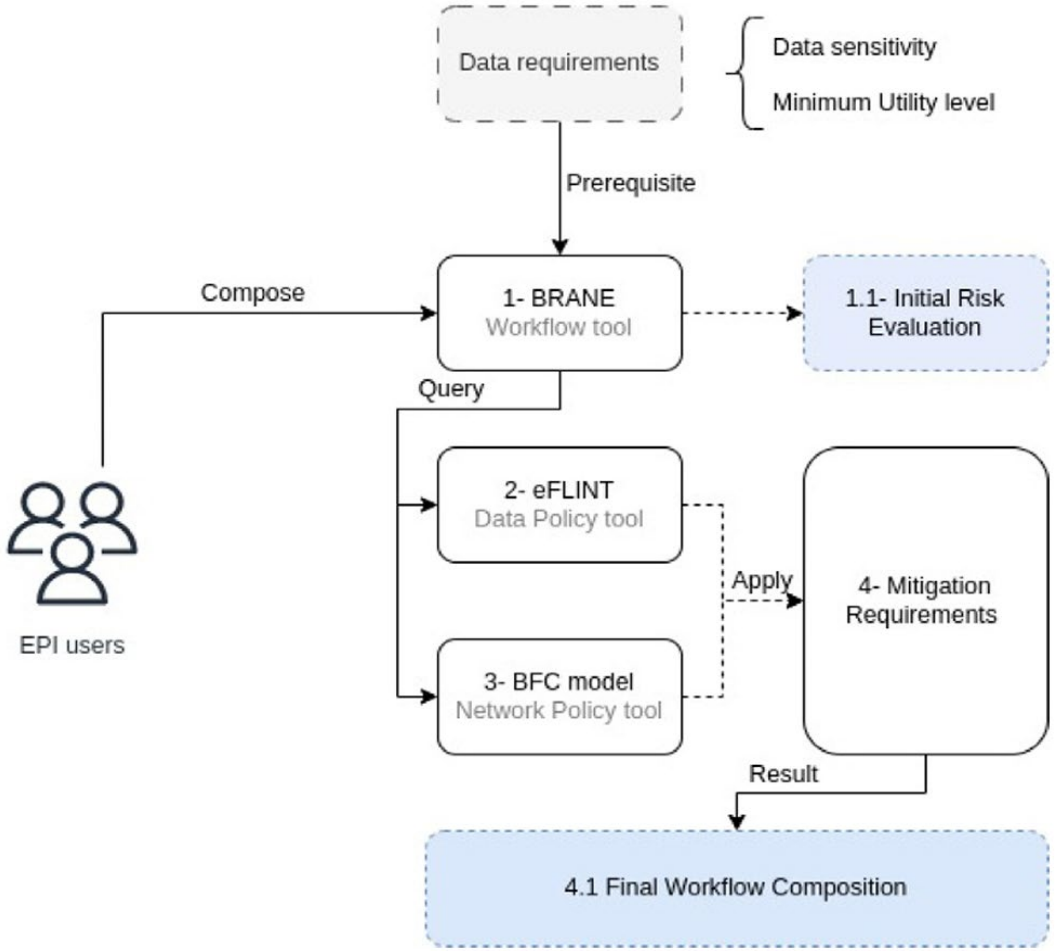
16:40-17:00 uur
Thomas van Binsbergen – Bringing theory to practice && Closing

Static ex-ante enforcement

Orchestration of workflow and middleware based on policy

Compliance-checking of system model against normative model

Simulation and model-checking of normative model behaviour

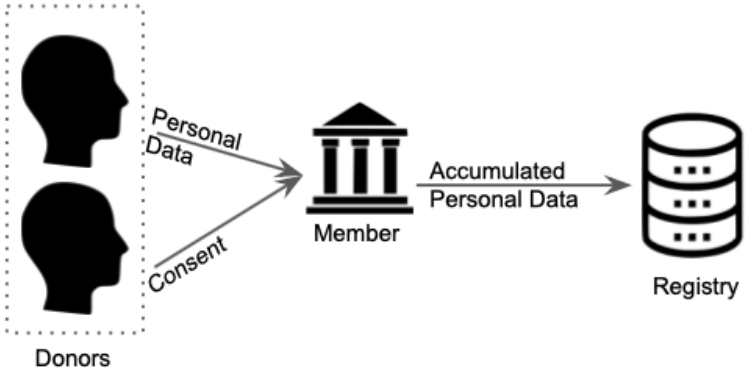


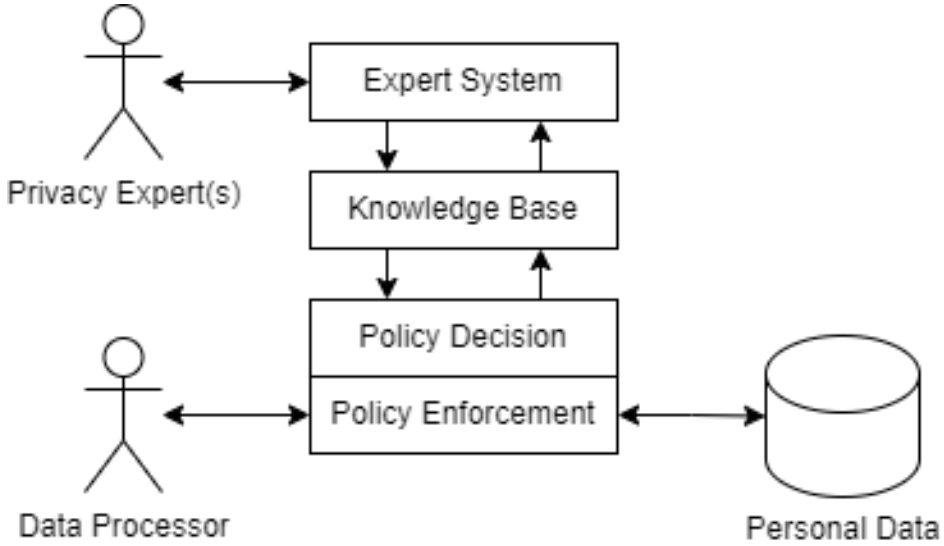
EPI framework orchestration; privacy by design

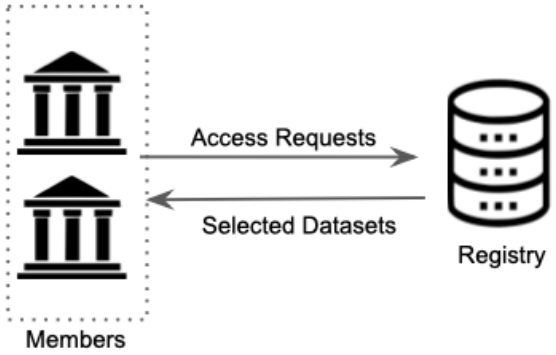
https://doi.org/10.1016/j.future.2024.107550

Dynamic ex-ante enforcement

Dynamic generation of **access control** policies from social policies
https://doi.org/10.1016/j.procs.2021.12.221



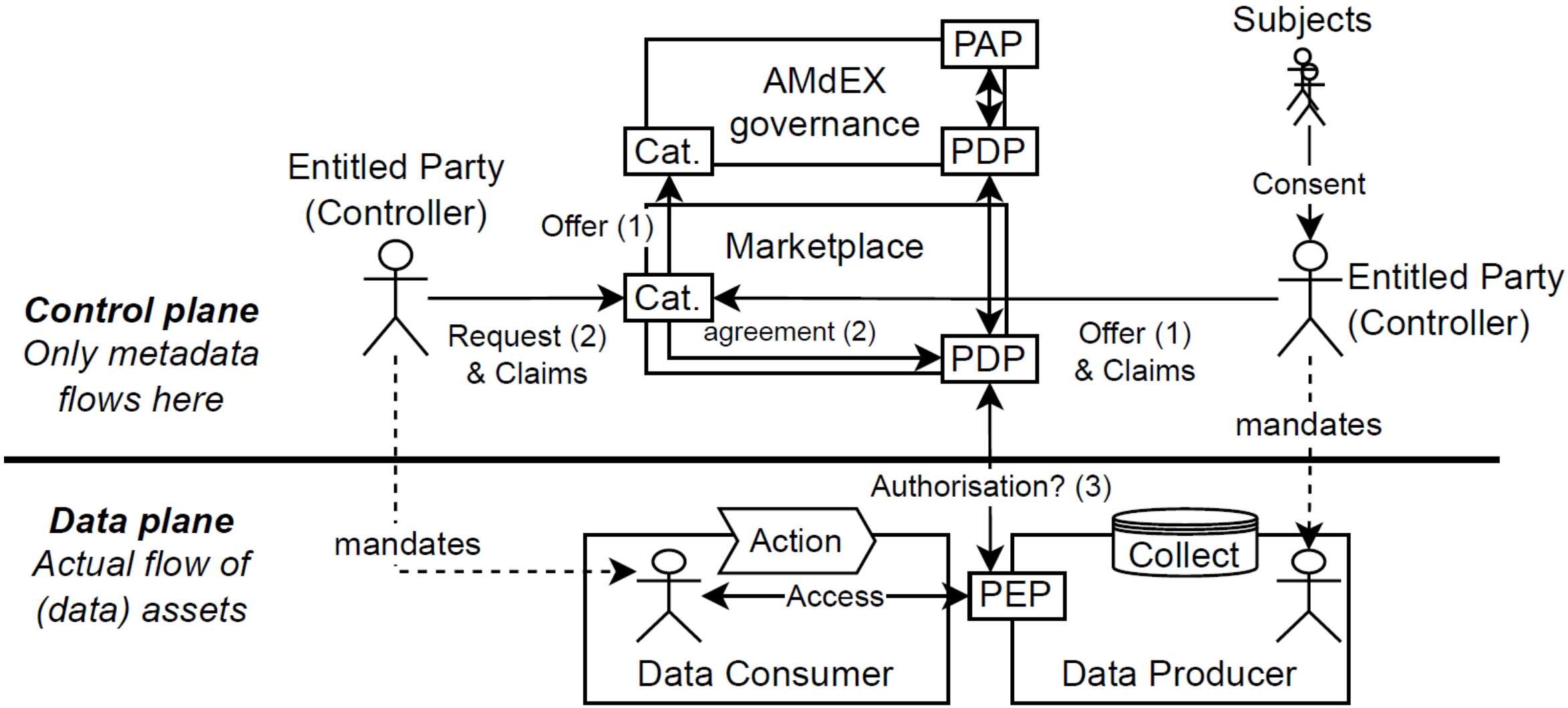




Lawful and Accountable Personal Data Processing with
GDPR-based Access and Usage Control
in Distributed Systems

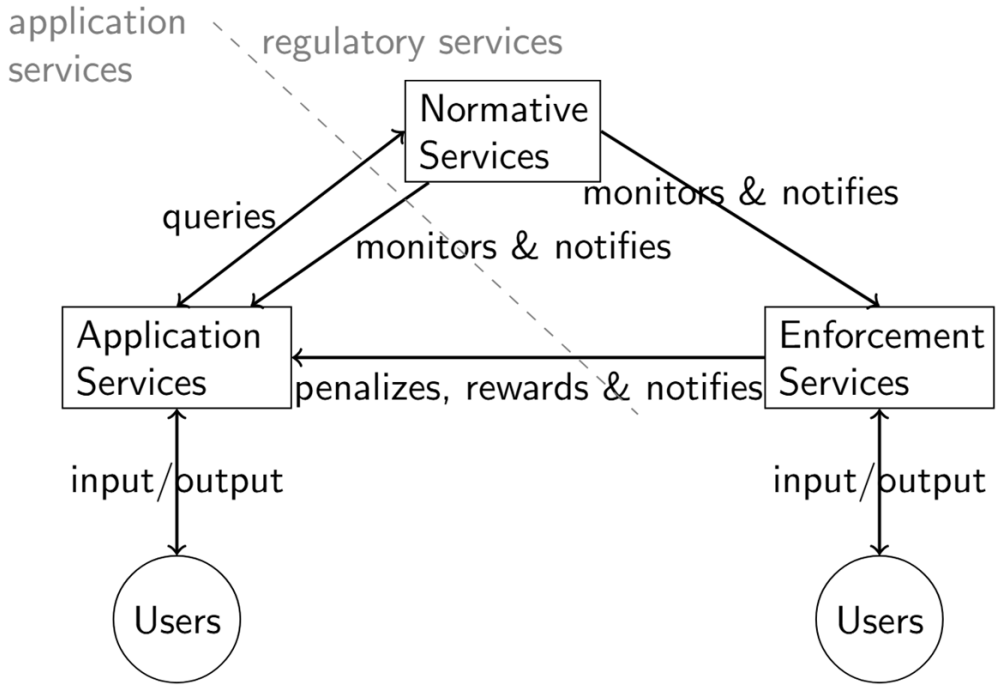
*Based on **XACML** architecture*

Integrating GDPR-based access control in DMI infrastructure



Dynamic ex-post enforcement

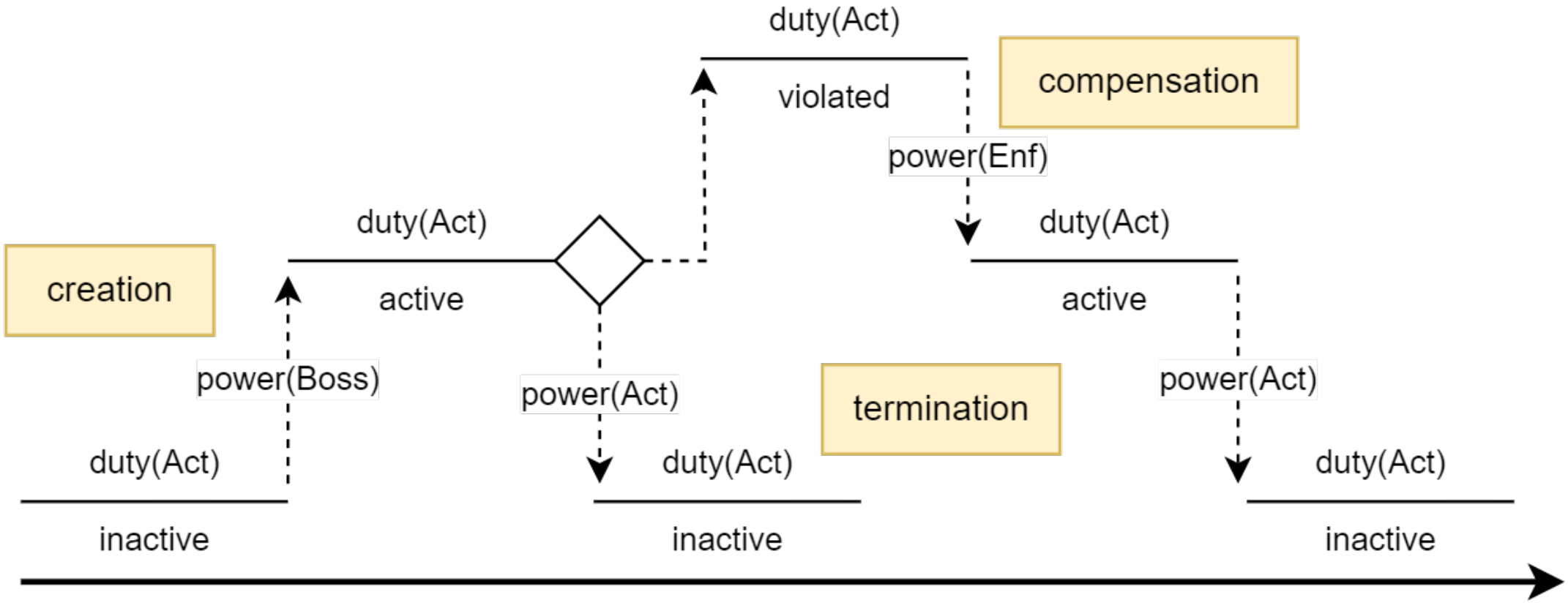
Regulatory Services to Automate Compliance with Ex-post Enforcement
https://ltvanbinsbergen.nl/files/papers/aicol-2023.pdf



Explicit **violations**

Enforcement of **legal obligations**
based on **timed events** and (internal and external) **monitors**
human involvement to provide facts, observations and judge violations

Duty lifetime

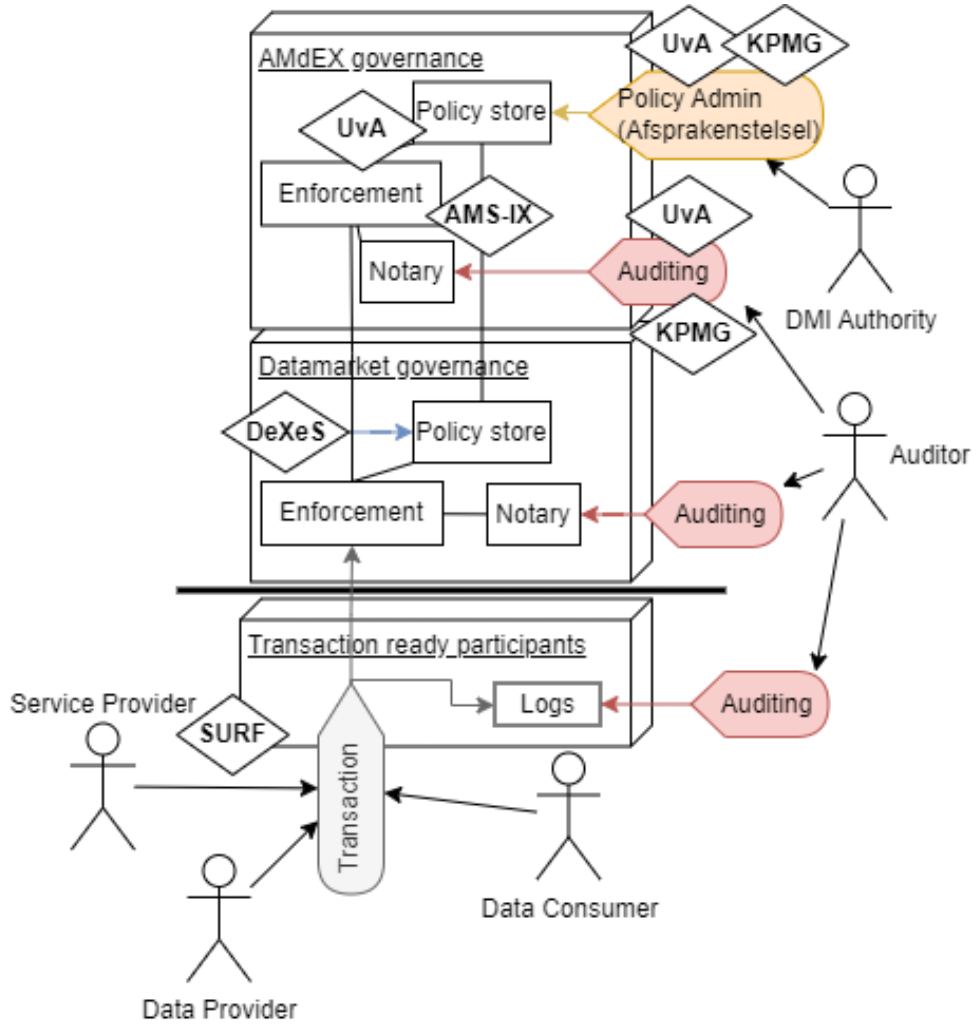


Static ex-post enforcement

Access to logs and compliance decisions (accountability)

Checking cases from logs for compliance (auditing)

Partially automated construction of audit trails → audit report → trust ranking



Main takeaways

We presented solution ingredients that enable **automating compliance** against governance rules (such as Afsprakenstelsel), regulations (such as GDPR) and usage conditions.

This achieved via reasoning with **formal specifications** (software code)

Reasoning is applied both *before* and *after* the fact.

And needs to be accountable and based on collected evidence, logs, etc.
Raising challenges regarding distribution of responsibilities and scalability

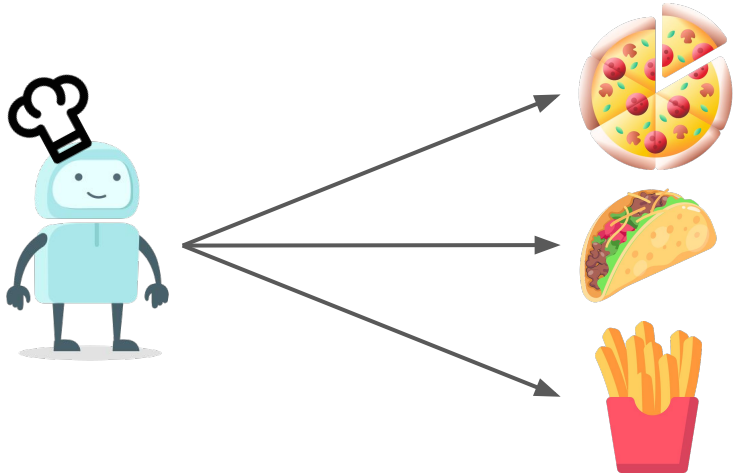
Executable policy: eFLINT & DMI Commons

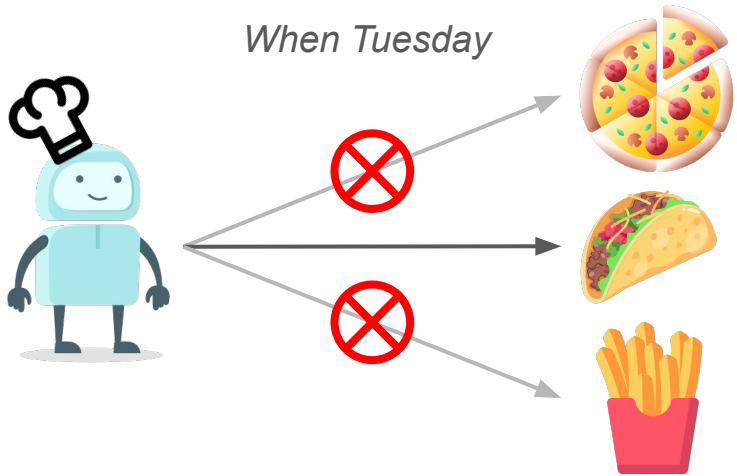
Tim Müller, University of Amsterdam

UvA-AMdEX research update, **21 May 2025**

So, we've heard about these "norms"...

- We see norms simply as **directives that constrain a system**
- I.e., some **actions** (or some **states**) are declared **undesirable**





"On Tuesdays, always make tacos"

So, we've heard about more "norms"...

- They can **vary greatly** and be **very complex**
- Examples:
  - "Don't spill your food!"
  - "Tuck your shirt in your pants"
  - "When you buy something at a shop, you have to pay"
  - "Teachers are allowed to dismiss class"
  - "Come to me if you disagree with each other"
- For this presentation, we mostly consider them from **legal sources**
  - Although in DMI, **policy templates** may capture *any* norm

DMI's norms

- The **Afsprakenstelsel**
- And in particular, **Article 7**
  - Defines **"Commons"** and what participants to the ecosystem are **permitted** to (not) do with them

Artikel 7 Commons

7.1 Producten, diensten en data binnen het DMI-ecosysteem die grotendeels zijn gefinancierd met publieke middelen, dienen om niet beschikbaar te zijn voor Deelnemers en zijn geen Commons. Kosten die ontstaan als gevolg van het daadwerkelijk gebruik van deze producten, diensten en data kunnen echter wel aan desbetreffende Deelnemer(s) worden doorberekend.

7.2 Iedere Deelnemer moet jaarlijks ten bedrage van een door de Ecoraad vast te stellen Eurowaarde eigen producten, diensten en/of data(gebruik) aan elk van de andere Deelnemers beschikbaar te stellen als Commons, even zoals iedere Deelnemer recht heeft om jaarlijks voor datzelfde vastgestelde bedrag producten, diensten en/of datagebruik af te nemen als Commons van elk van de andere Deelnemers.

7.3 Voor 2023 bedraagt de vastgestelde Eurowaarde voor de Commons per Deelnemer Euro 2.500,-- (inclusief BTW) op jaarbasis. Deze Eurowaarde moet direct herleidbaar zijn tot de commerciële waarde van de als Commons aangeboden producten, diensten en/of data(gebruik) in de Producten en Diensten Catalogus.

7.4 Levering en afname van Commons dient altijd reguliere, in de Producten en Diensten Catalogus opgenomen en voor levering als Commons aangemerkte producten, diensten en data(gebruik) te betreffen. Afhandeling van de levering en afname ervan dient altijd plaats te vinden via de Algemene Voorzieningen.

7.5 Levering van producten, diensten en data(gebruik) als Commons moet altijd vrij van kosten zijn jegens de leverende Deelnemer

7.6 Om kosten van in te kopen data(gebruik) zo laag mogelijk te kunnen houden voor kennisinstituten en onderzoeksinstellingen, dienen alle Deelnemers aan dezen data(gebruik) altijd aan te bieden als Commons, dus vrij van kosten. Indien er inspanningen verbonden zijn aan de distributie en/of bewerkingen van die data, dan mogen de redelijke kosten daarvan bij de ontvanger in rekening worden gebracht. De kennis die door desbetreffende kennisinstituten en onderzoeksinstellingen mede door deze als Commons verstrekte data wordt opgebouwd, dient door dezen altijd kosteloos beschikbaar te worden gesteld aan alle Deelnemers.

7

https://dmi-ecosysteem.nl/wp-content/uploads/bb_documents/2023/10/2023.06.01-DMI-Afsprakenstelsel-v1.pdf

DMI’s norms, automated

- Now we want to **enforce the Afsprakenstelsel automatically**
- The central question of this presentation is:

  *“How can we **automate** the **enforcement** of DMI’s **norms**?”*

- I.e., how can we make the Afsprakenstelsel **executable**?

The meat and potatoes

- Example: turn Article 7.5 into an **executable norm**

7.5 Levering van producten, diensten en data(gebruik) als Commons moet altijd vrij van kosten zijn jegens de leverende Deelnemer

-or-

7.5 *"Delivery of products, services and data(usage) as Commons must always be free of costs for the supplying Participant"*

The meat and potatoes - Executable norms

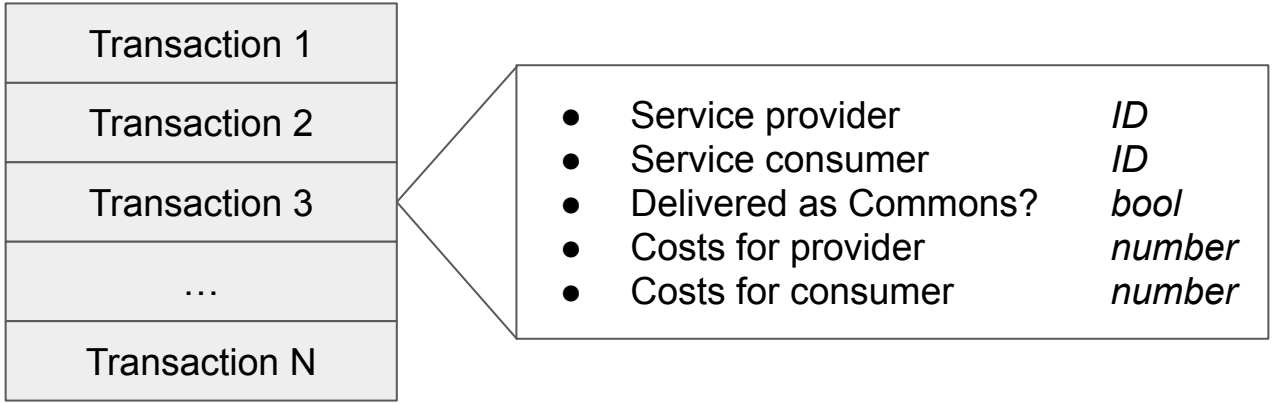
- We can write down norms **computationally** (i.e., as little programs)
- Then we can use them to **validate** a particular **system state**

Hence, we need to:

1. **Model** a system state;
2. **Formalise** our norm as a function; and
3. **Apply** the function to a particular state.

> ***7.5*** *"Delivery of products, services and data(usage) as Commons must always be free of costs for the supplying Participant"*

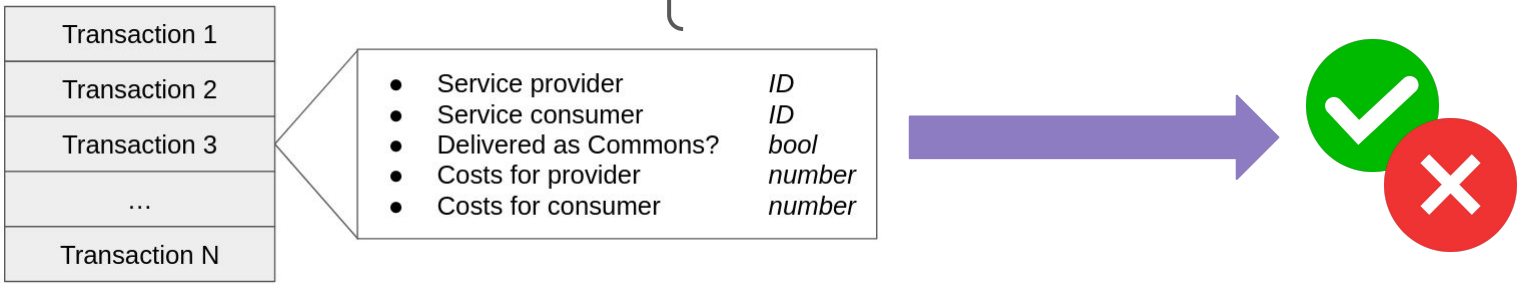
The meat and potatoes - 1. The model



7.5 *"Delivery of products, services and data(usage) as Commons must always be free of costs for the supplying Participant"*

The meat and potatoes - 2. The function

IF Delivered as Commons? == True
THEN *assert* Cost for provider == 0

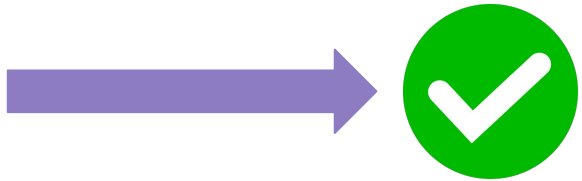


7.5 *"Delivery of products, services and data(usage) as Commons must always be free of costs for the supplying Participant"*

The meat and potatoes - 3. The application

IF Delivered as Commons? == True
THEN *assert* Cost for provider == 0

- Service provider — **University of Amst.**
- Service consumer — **Amy's Data World**
- Delivered as Commons? — **No**
- Costs for provider — **20**
- Costs for consumer — **50**

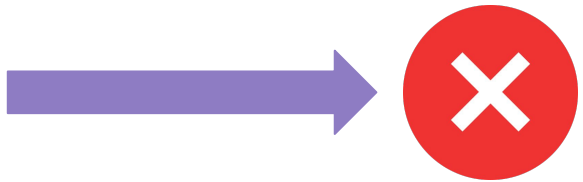


7.5 *"Delivery of products, services and data(usage) as Commons must always be free of costs for the supplying Participant"*

The meat and potatoes - 3. The application

IF Delivered as Commons? == True
THEN *assert* Cost for provider == 0

• Service provider	**University of Amst.**
• Service consumer	**Bob Analytics**
• Delivered as Commons?	**Yes**
• Costs for provider	**20**
• Costs for consumer	**50**



7.5 *"Delivery of products, services and data(usage) as Commons must always be free of costs for the supplying Participant"*

eFLINT - Normative specification

- eFLINT is a **language** for **formalising** all three
- Based on **Hohfeld's legal framework**
- **Model** the world as legal **powers**, **obligations**, **permissions**, …
- Specifically,
  - Express the world as *possible* **facts**, **actions** and **duties**
  - Express norms as **conditions** on **actions** & **duties**
  - Instantiate a model by describing which **facts**, **actions** and **duties** are really there.

```
Fact agent Identified by String.
Fact food Identified by Pizza, Tacos, Fries.

Fact day-of-the-week Identified by Monday, Tuesday, Wednesday,
                                   Thursday, Friday, Saturday,
                                   Sunday.
Fact current-day-of-the-week Identified by day-of-the-week.

Act cook
    Actor agent
    Related to food
    Derived from (Foreach agent, food: cook(agent, food))
    Conditioned by (Forall current-day-of-the-week:
        current-day-of-the-week.day-of-the-week != Tuesday
        || food == Tacos
    ).
```

The meat and potatoes - eFLINT

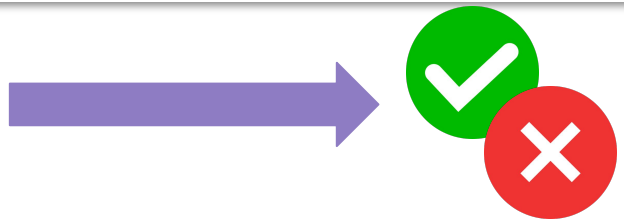
- We instead model the world as…
- …possible **transactions**
- …**participants** having **powers** to do **transactions** (they can take an action)
- Then we formalize norms as…
- …**conditions** on actions marking some as **violating**
- …**duties** modelling **obligation to transact in certain ways** for participants
- Finally, we apply a scenario by…
- …**instantiating** the model for a **specific transaction** or other scenario

> ***7.5*** *"Delivery of products, services and data(usage) as Commons must always be free of costs for the supplying Participant"*

The meat and potatoes - 1. 2. eFLINT

```
/// Dit kan verschillen van de algemenere `aanbod-kosten` omdat gebruikers onderling
/// kunnen afspreken.
Fact overeenkomst-kosten Identified by overeenkomst * waarde.
Extend Fact waarde
    Derived from (Foreach overeenkomst-kosten : overeenkomst-kosten.waarde).

/// Specificeert hoeveel gebruikskosten er aan een overeenkomst hangen voor een bepaalde Deelne
///
/// Dit kan verschillen van de algemenere `aanbod-gebruiks-kosten` omdat gebruikers onderling
/// andere kosten kunnen afspreken.
Fact overeenkomst-gebruiks-kosten Identified by overeenkomst * waarde.
Extend Fact waarde
    Derived from (Foreach overeenkomst-gebruiks-kosten : overeenkomst-gebruiks-kosten.waarde).

/// Specificeert hoeveel gebruikskosten er verbonden zijn aan het gebruik van de AV per overeer
Fact overeenkomst-av-kosten Identified by overeenkomst * waarde.
```

```
Duty commons-vrij-van-kosten
    Holder aanbieder
    Claimant beheerder
    Related to overeenkomst
    // Deze plicht bestaat voor alle aanbieders die Commons aanbieden
    Derived from (Foreach overeenkomst, beheerder :
        commons-vrij-van-kosten(overeenkomst.aanbod.aanbieder, beheerder, overeenkomst)
        When aanbod-als-commons(overeenkomst.aanbod)
    )
    // Deze plicht wordt geschonden als er kosten worden doorgerekend
    Violated when (Exists overeenkomst-kosten :
        overeenkomst-kosten.overeenkomst == overeenkomst
        && overeenkomst-kosten.waarde > 0
    ).
```

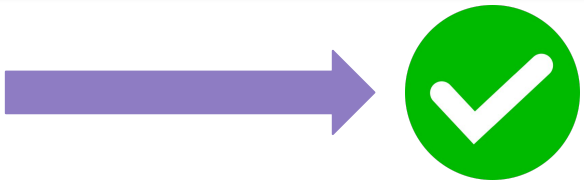


7.5 *"Delivery of products, services and data(usage) as Commons must always be free of costs for the supplying Participant"*

The meat and potatoes - 3. eFLINT

```
Duty commons-vrij-van-kosten
    Holder aanbieder
    Claimant beheerder
    Related to overeenkomst
    // Deze plicht bestaat voor alle aanbieders die Commons aanbieden
    Derived from (Foreach overeenkomst, beheerder :
        commons-vrij-van-kosten(overeenkomst.aanbod.aanbieder, beheerder, overeenkomst)
        When aanbod-als-commons(overeenkomst.aanbod)
    )
    // Deze plicht wordt geschonden als er kosten worden doorgerekend
    Violated when (Exists overeenkomst-kosten :
        overeenkomst-kosten.overeenkomst == overeenkomst
        && overeenkomst-kosten.waarde > 0
    ).
```

```
+dienst-waarde(Appel, 1000).
+aanbod(0, Amy, Appel, 2023).
+aanbod-als-commons(aanbod(0, Amy, Appel, 2023)).
+overeenkomst(aanbod(0, Amy, Appel, 2023), Bob).
+overeenkomst-kosten(overeenkomst(aanbod(0, Amy, Appel, 2023), Bob), 0).
```



7.5 *"Delivery of products, services and data(usage) as Commons must always be free of costs for the supplying Participant"*

What we're doing, *really*

- Is **running** an executable norm the same as **judgement**?
  - We *are* **categorizing** things as **compliant** or **non-compliant**, after all

- However…
  - We are always doing categorization based on **models**
  - This is an allegory for our **understanding** of a situation
  - **Judgement is also about checking if one's understanding is correct**

- Hence:
  - We've only done **legal derivation** (is my model compliant?)
  - Something else needs to do the **qualification** (instantiating the model accurately)

Thanks for listening!



DMI·ECOSYSTEEM

- We’re interested into making **norms** executable
- We can **formalize** them as **executable norms**
- Which can **automate legal reasoning** of a **norm** on a **particular model instantiation**
- We use **eFLINT** to program the reasoning
- **Qualification** is done **externally**, mapping the **real system** to the **model**
  - Up to a sense, this is always human, case-by-case work

t.muller@uva.nl
https://gitlab.com/eflint
https://definities.dmi-ecosysteem.nl

UvA Research Update

Dataspace Governance

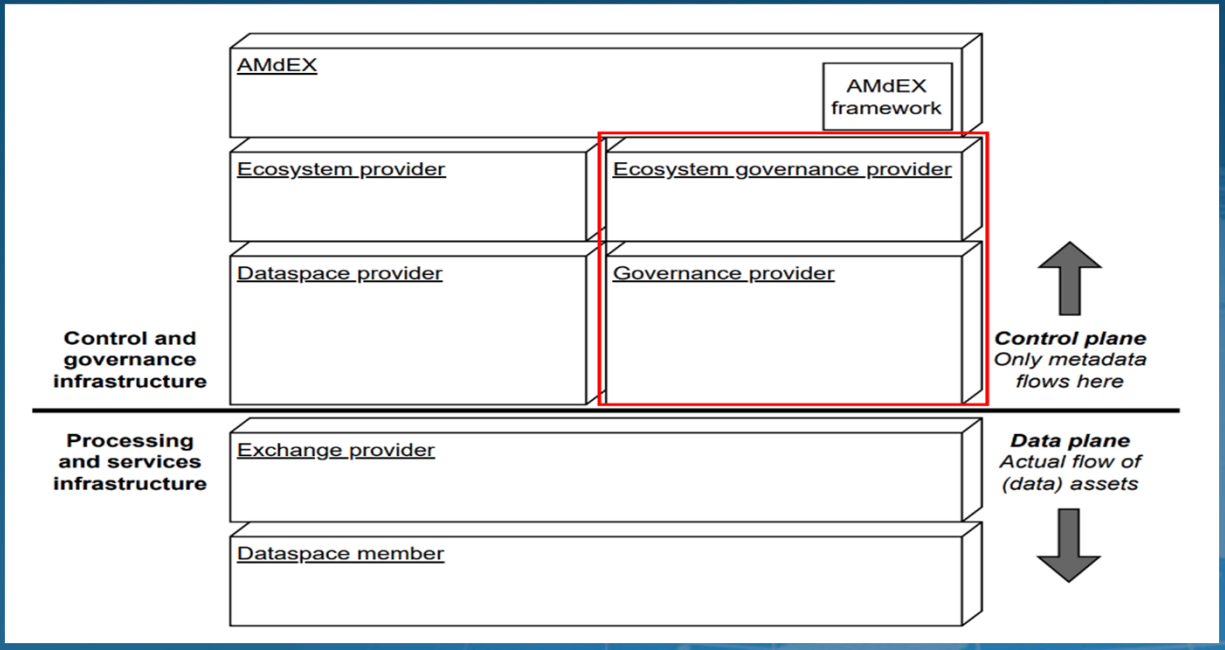
Policy registration and information dossiers

Merrick Oost-Rosengren
m.a.oost@uva.nl

AMdEX

DMI·ECOSYSTEEM

Governance

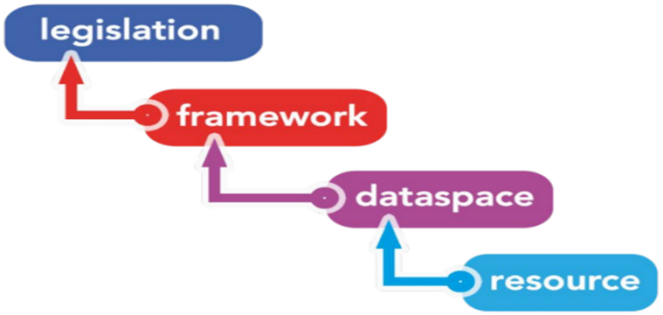


AMdEX Reference Architecture

Governance within Common European Data Spaces

Governance within a common European data space should be based on principles that closely relate to the general values (or pillars) on which the EU was founded

Governance manages legal obligations



Rule of law,
International, EU and local

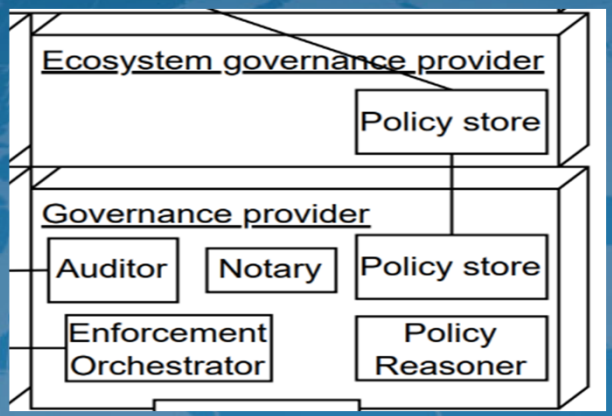
Trust eco-system & governance
principles for sharing data

Consortium agreements
"how we share data"

Conditions for sharing
specific data, services,
documents, applications

Enforcement Orchestrator

- Creates a dossier with meta-data for legal obligations
- Uses the Policy store and Policy Reasoner to validate the legal obligations
- Use the Notary to store mutations on dossiers
- Assists the Auditor during auditing



Dossier

- Contains one or more agreements governing one data sharing instance
- May contain one or more applicable laws
- Contains signatures of the parties or delegates involved
  - Within DMI a datamarket has been mandated to sign
- Contains additional data related to the agreement, not containt on the agreement.

Clearing (5)

- During clearing the dossier is created
- If a dossier contains machine executable agreements, these are validated.
- A human readable version of the agreement is returned to the Datamarket (DMI)
- The Datamarket signs the agreement on behalf of the parties (DMI)

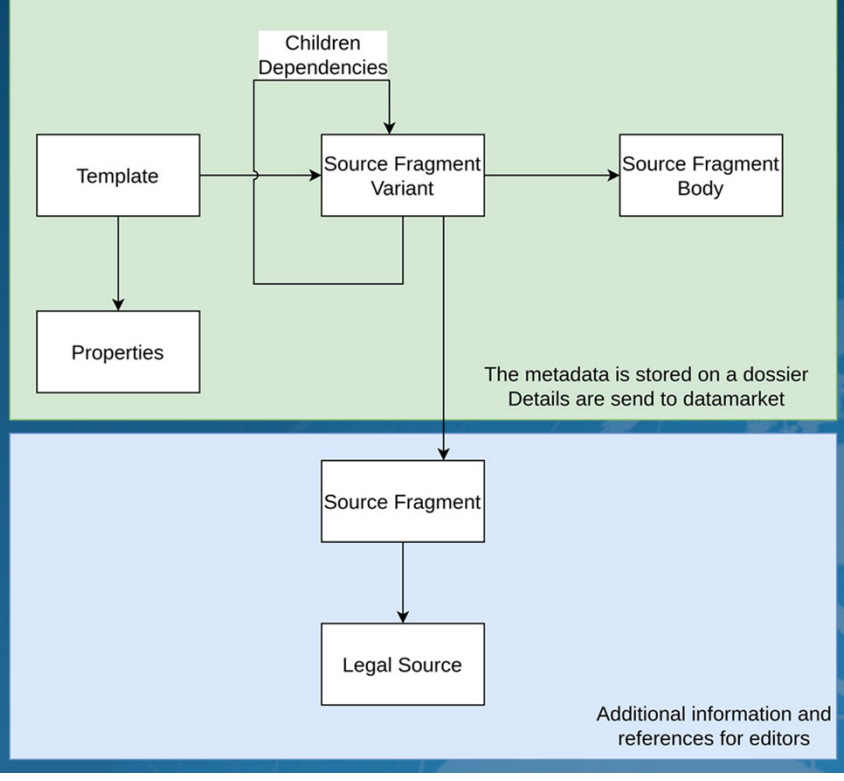
Policy Store

- Contains agreements/policies/laws in different languages (dutch legal, english legal, synopsis, machine readable/executable)
- The different languages of the same agreement are connected.
- It returns the version of an agreement/policy/law for a specific date
  - A signable version, which only contains the meta-data of the agreement/policy/law
  - A full version, which contains the version in a specific language

Policy Store cont.

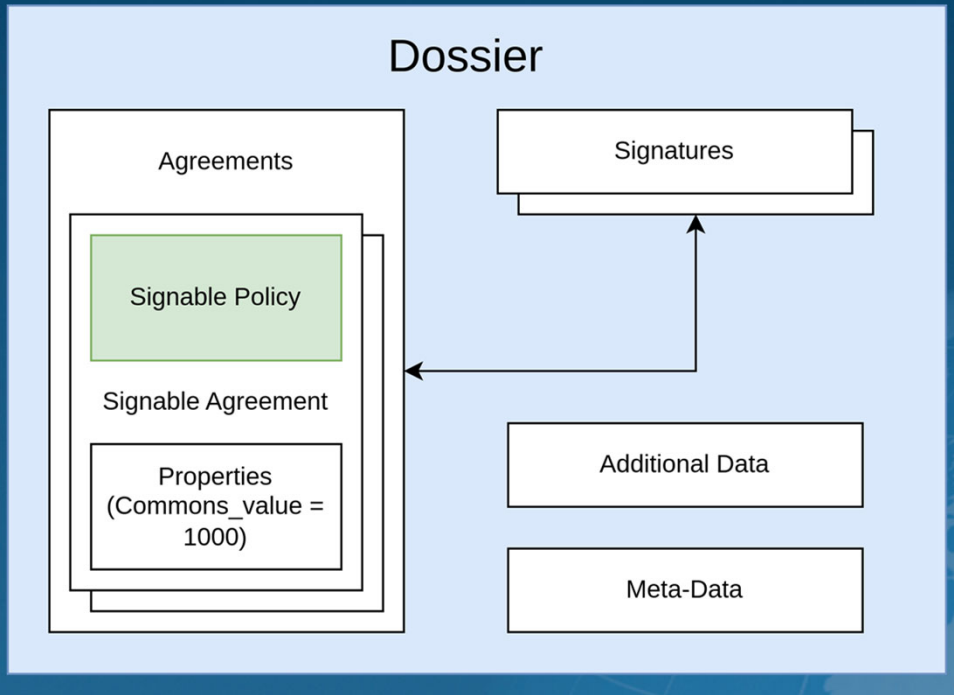
- The data in the policy store is immutable
- Agreements/Policies/Laws are updated by adding the new data to the policy store
- There may be multiple variants of the same legal text (e.g. higher level or details). There is always only one active for an agreement (per language).

Policy store data model



Policy Meta-Data / Signable Policy


```
"Template":"Commons",

"Language":"EFlint",

"Timestamp" : "21-05-2025",

"Fragments": [

  {

     "Fragment_ID" : "xyz",

     "Body_hash" : "avc-abx",

     "Dependencies" : [ { ... }, { ... } ]

     "Children" : [ { ... }, { ... } ]
```



Datamarket input:

"EntitledParty": "EU.EORI.EP",

"DataProvider": "EU.EORI.DP",

"DataConsumer": "EU.EORI.DC",

"ResourceId": "13",

"TransactionId": "12",

"Policies": [{

"PolicyTemplateId": "Commons",

"Parameters": [{

"name": "commons_value",

"value": "1000"

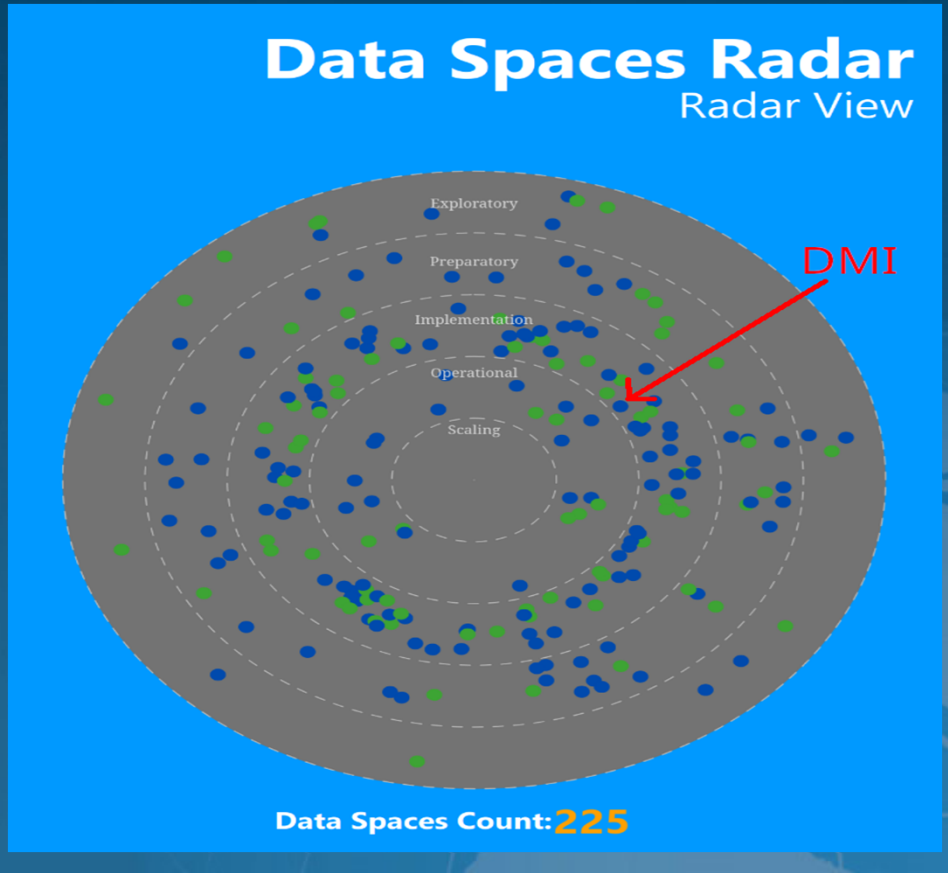
}] }

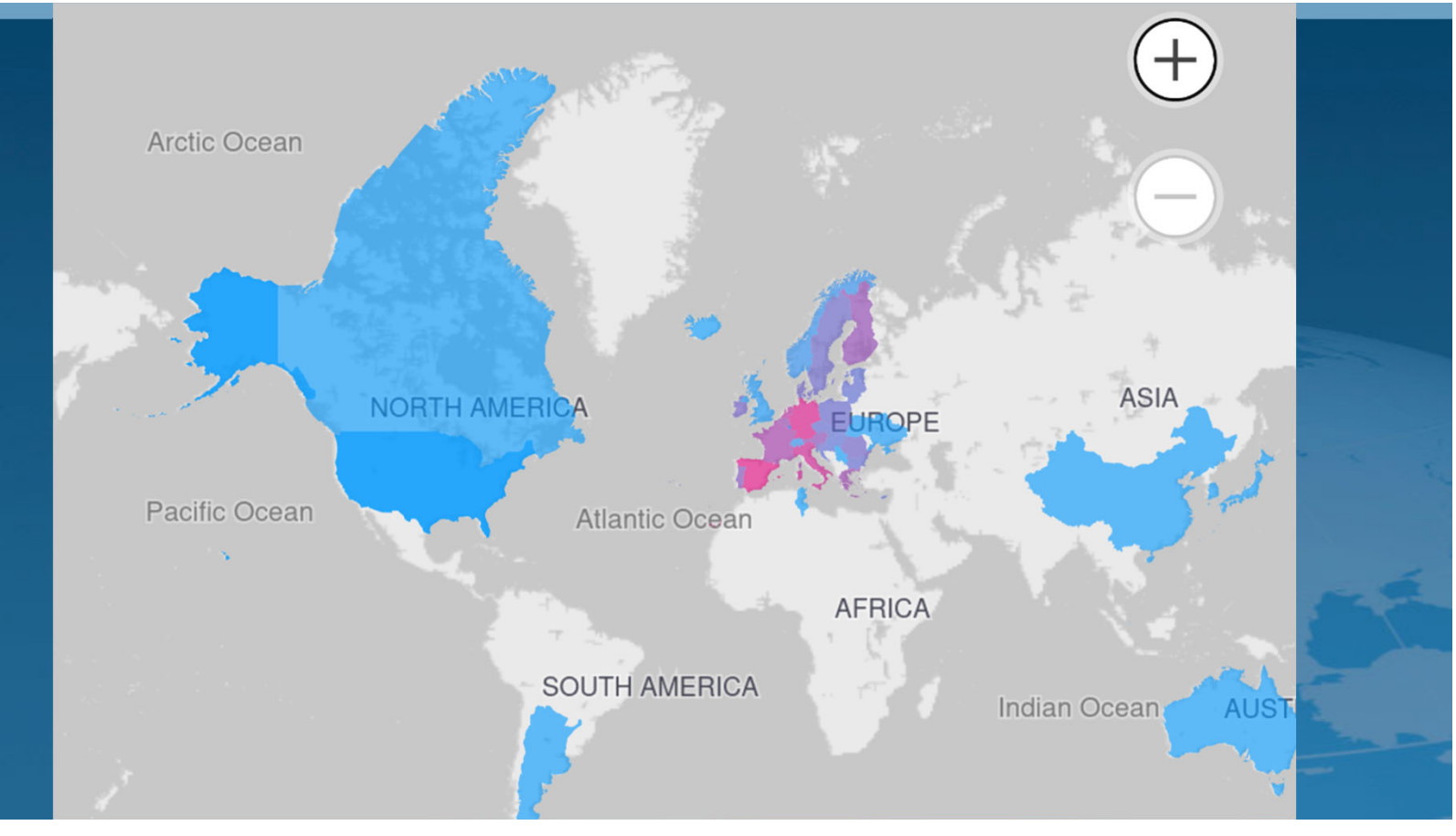
Governance Overview

- Handles the legal documentation of an transaction
- Checks if an agreement is compliant during creation
- Can check if the agreement is still compliant during access to data
- Can be used for (Legal) Audit
- Can supply the dossier during a dispute
  - For DMI, input from the Datamarket is also required, since the Datamarket signs

Additional functionality of governance (not used by DMI)

- Adjust agreements based on the requirements/allowances of the requesting and offering party
- Verify legal requirements when data moves accross a dataspace or country boundary


Data Spaces Radar
Radar View
Exploratory
Preparatory
Implementation
Operational
Scaling
DMI
Data Spaces Count: 225


Arctic Ocean
NORTH AMERICA
EUROPE
ASIA
Pacific Ocean
Atlantic Ocean
AFRICA
SOUTH AMERICA
Indian Ocean
AUST

References

- AmdEX Reference Architecture
  - https://zenodo.org/records/10565916
- Dataspace Radar
  - https://www.dataspaces-radar.org/radar/
- Governance within Common European Data Spaces
  https://wikis.ec.europa.eu/spaces/jrcdataspaceswiki/pages/78709328/3.4.+Governance+within+Common+European+Data+Spaces

Contact

Merrick Oost-Rosengren
m.a.oost@uva.nl

Policy Specification and Communication in Distributed Systems

Christopher Esterhuyse @ 21 May Research Update

University of Amsterdam
Informatics Institute

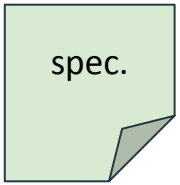
AMdEX

DMI·ECOSYSTEEM

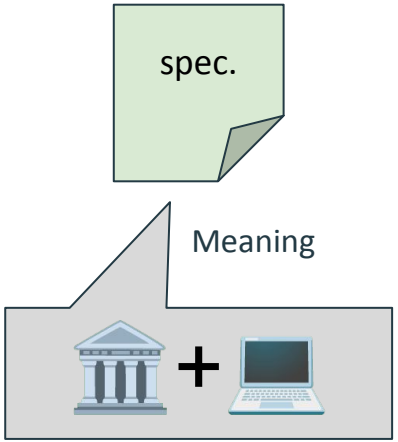
Contents

1. Vision: Specification-Centric Systems
2. Challenge: Distributing Everything
3. Approach: JustAct Framework
    - Idea: distributed specification + universal accountability
    - Examples from a case study

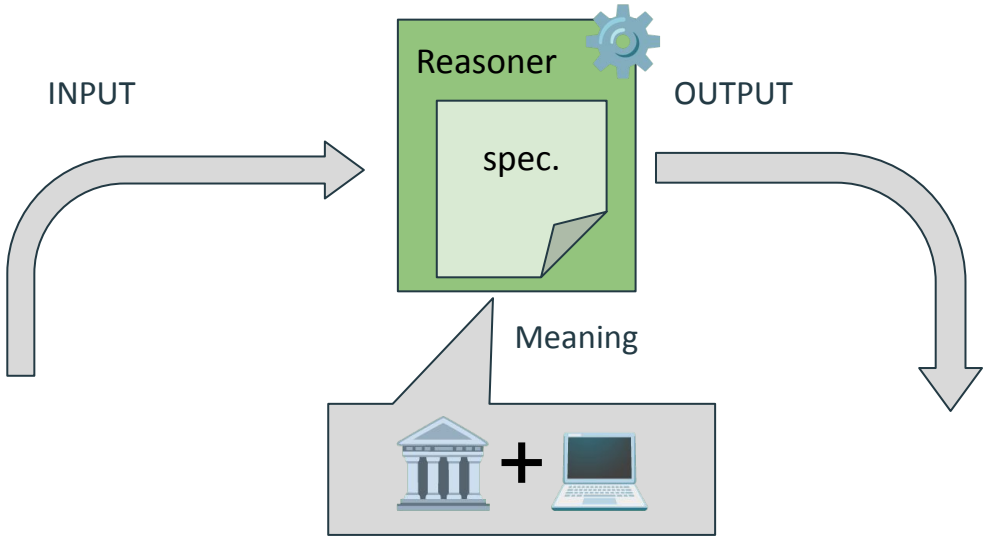
Vision: Specification-Centric Systems



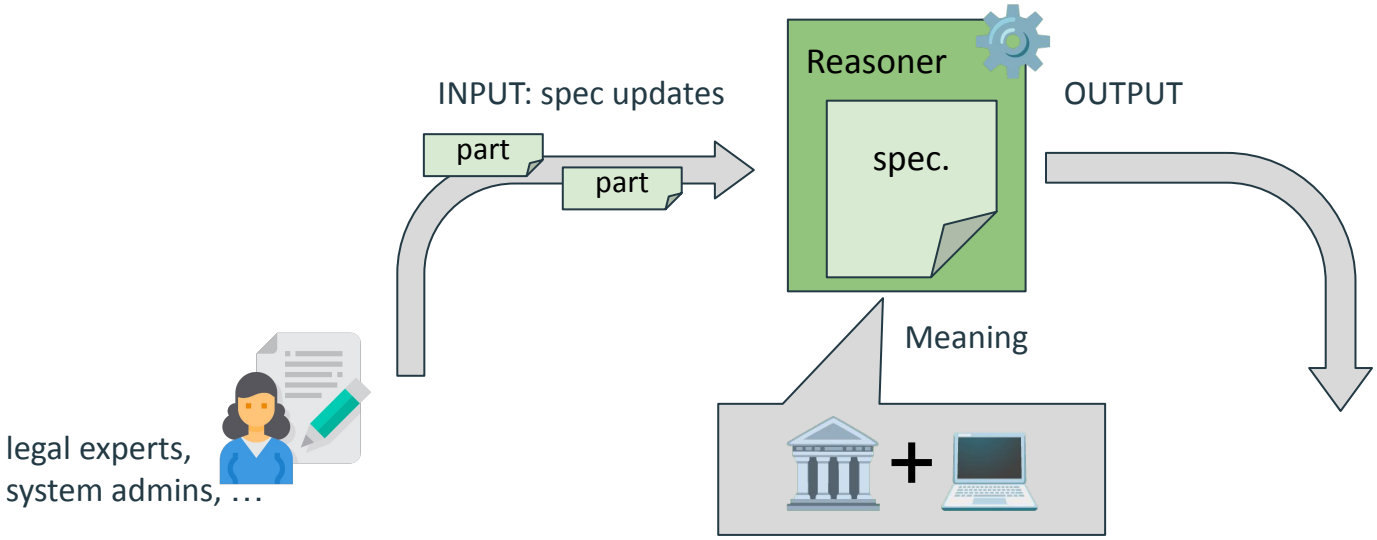
Vision: Specification-Centric Systems



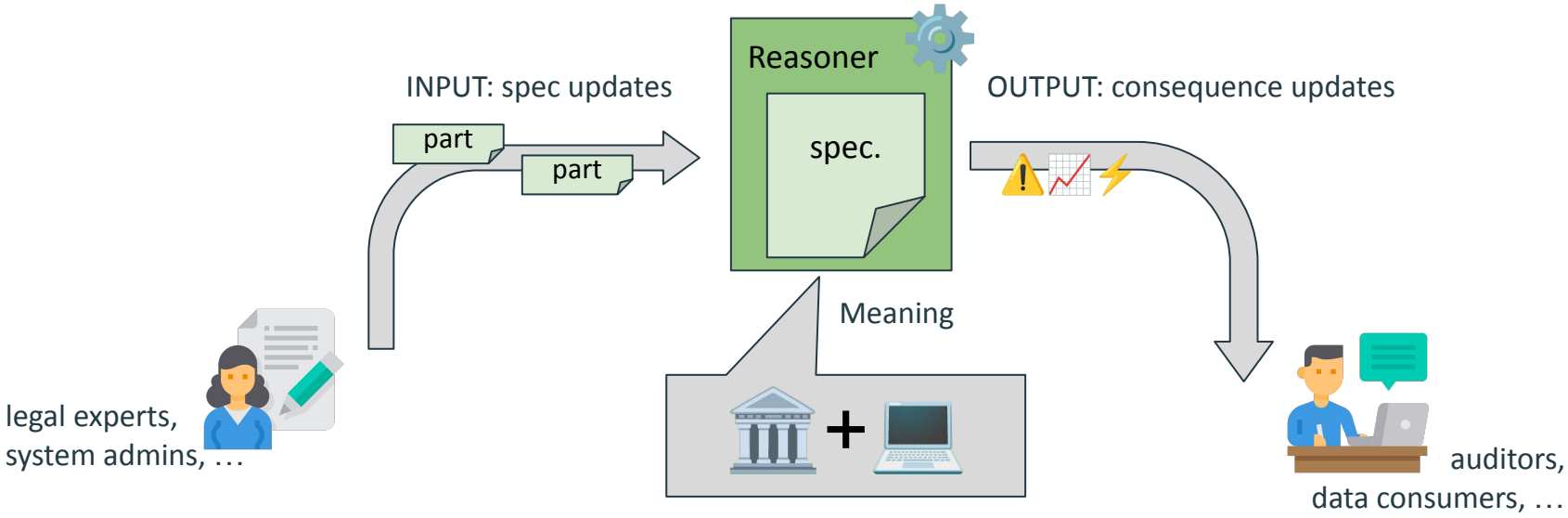
Vision: Specification-Centric Systems



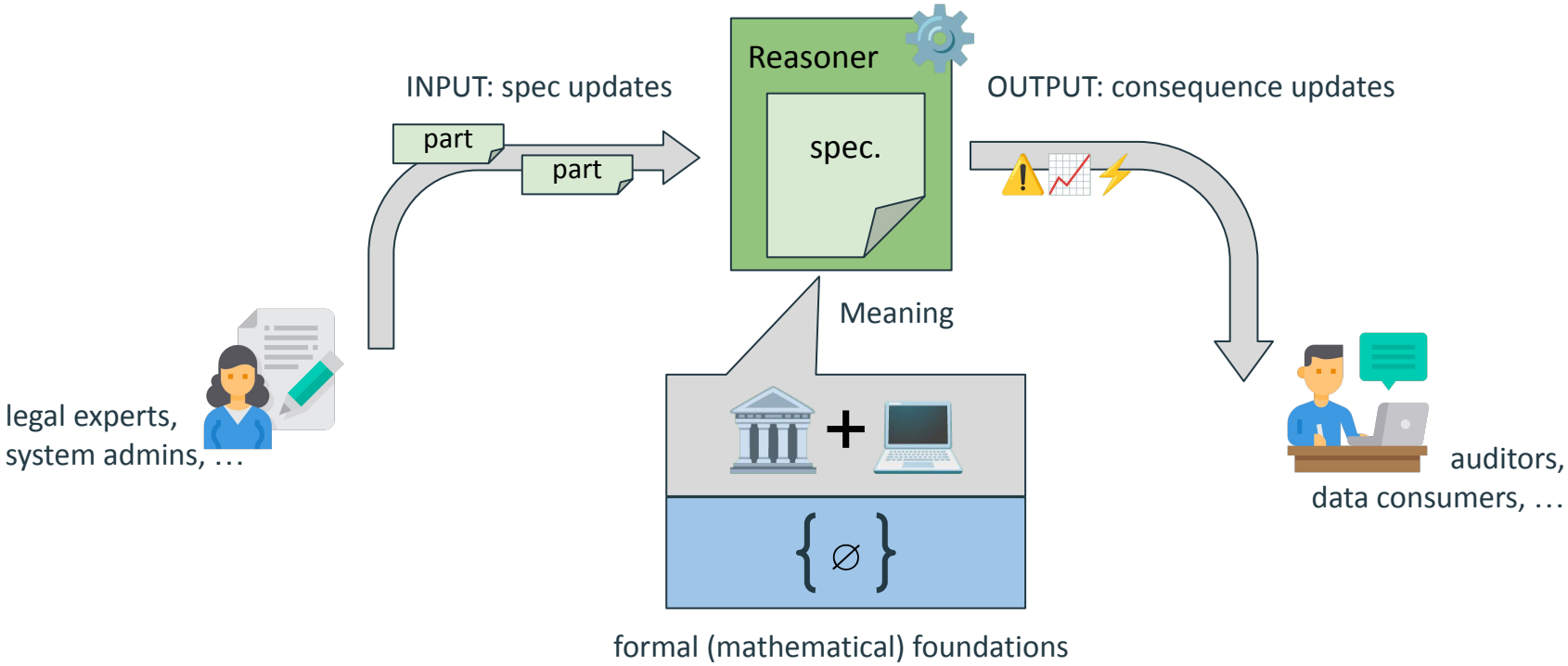
Vision: Specification-Centric Systems



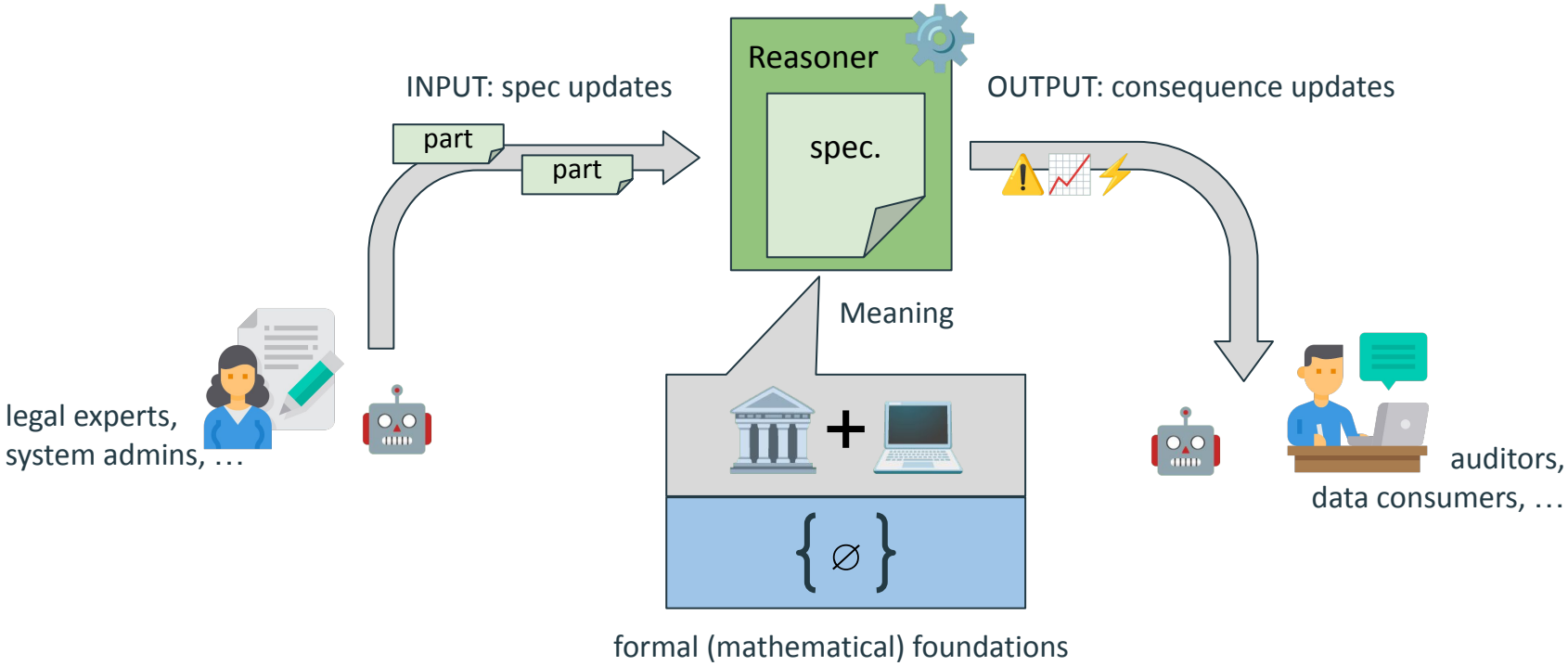
Vision: Specification-Centric Systems



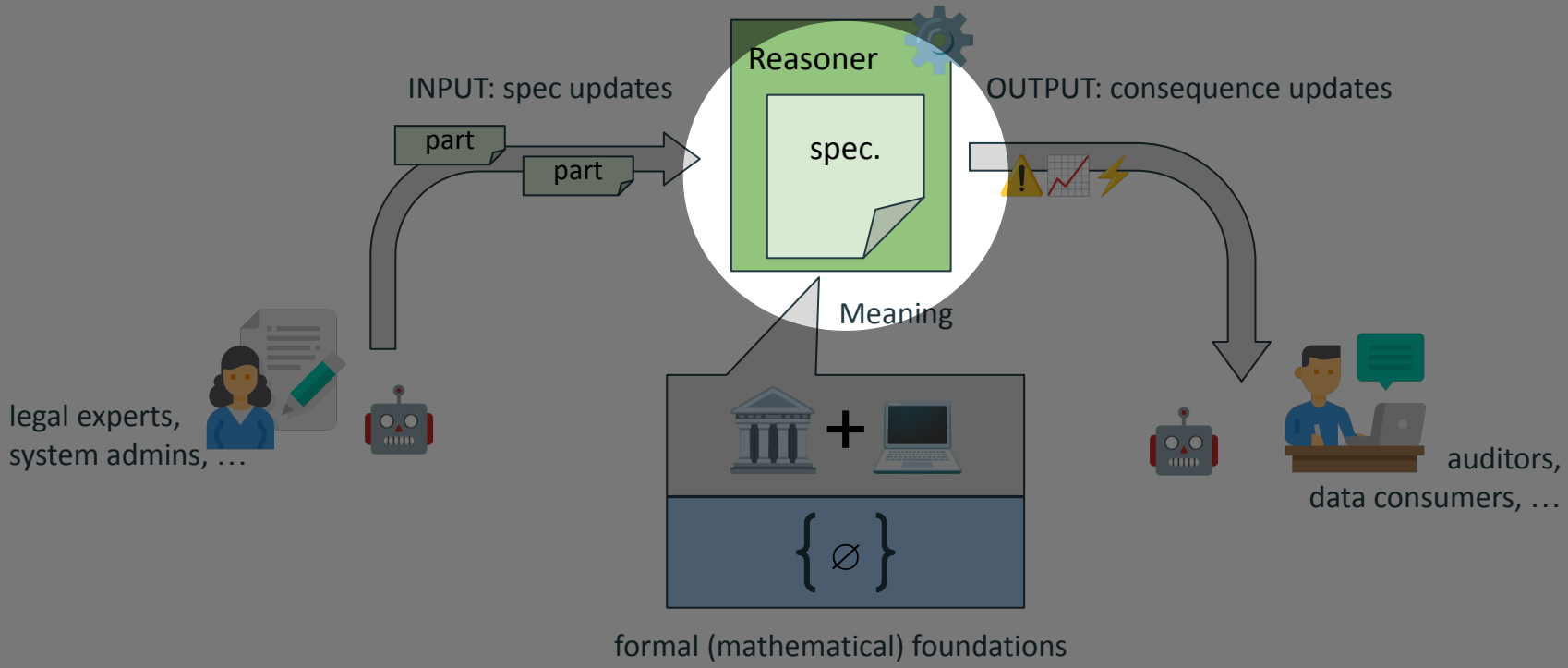
Vision: Specification-Centric Systems



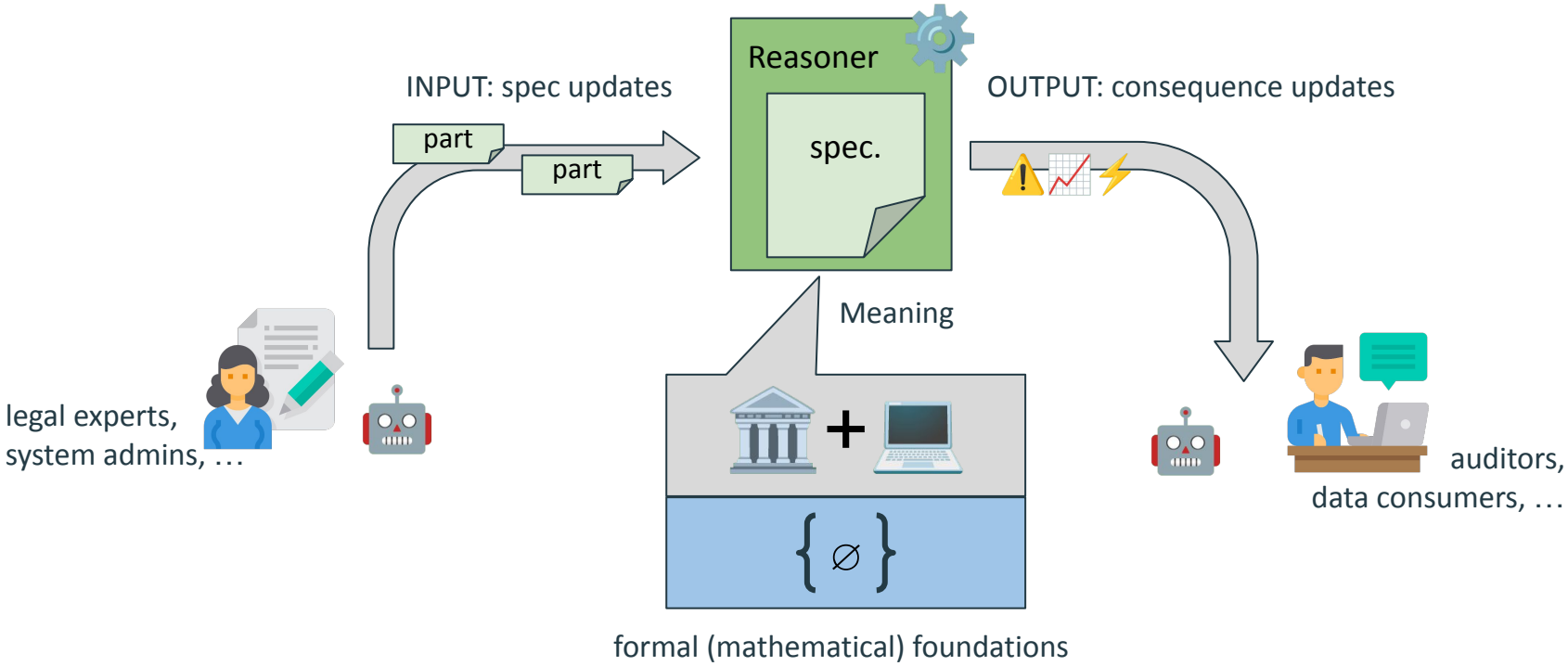
Vision: Specification-Centric Systems



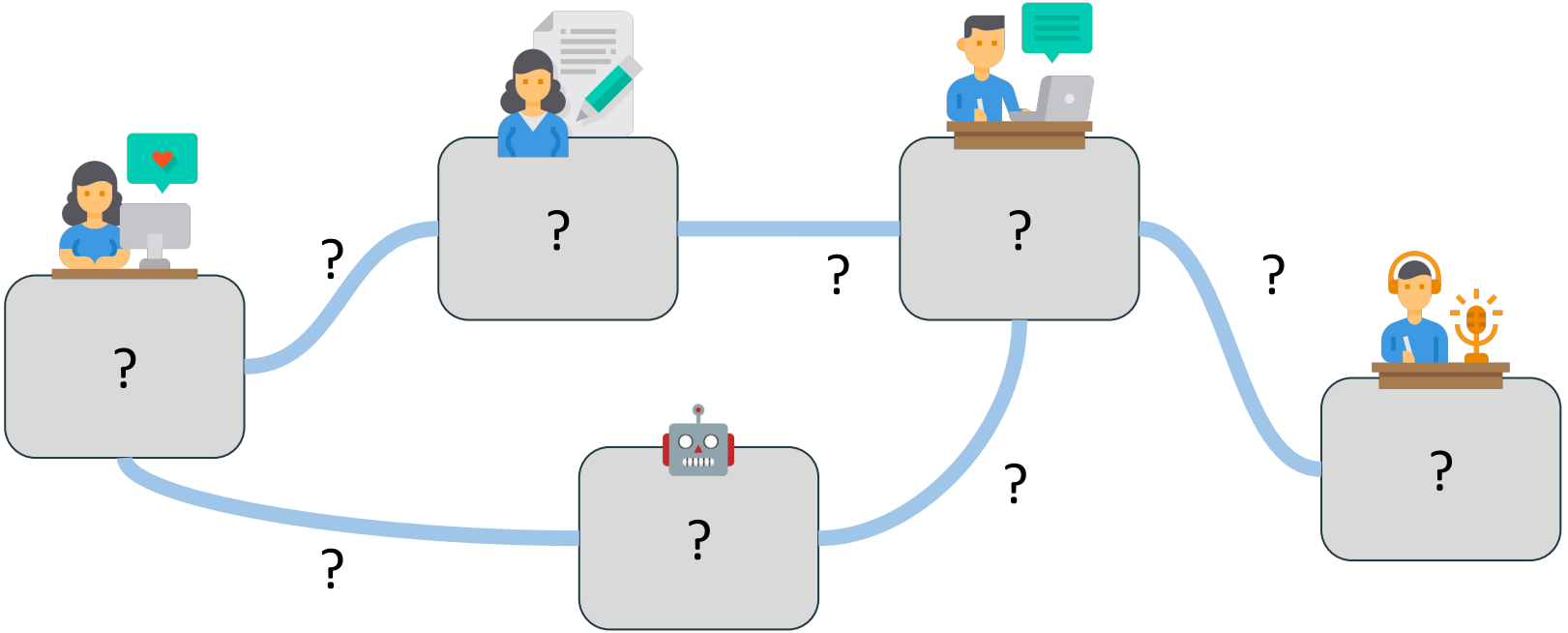
Vision: Specification-Centric Systems



Challenge: Distributing Everything

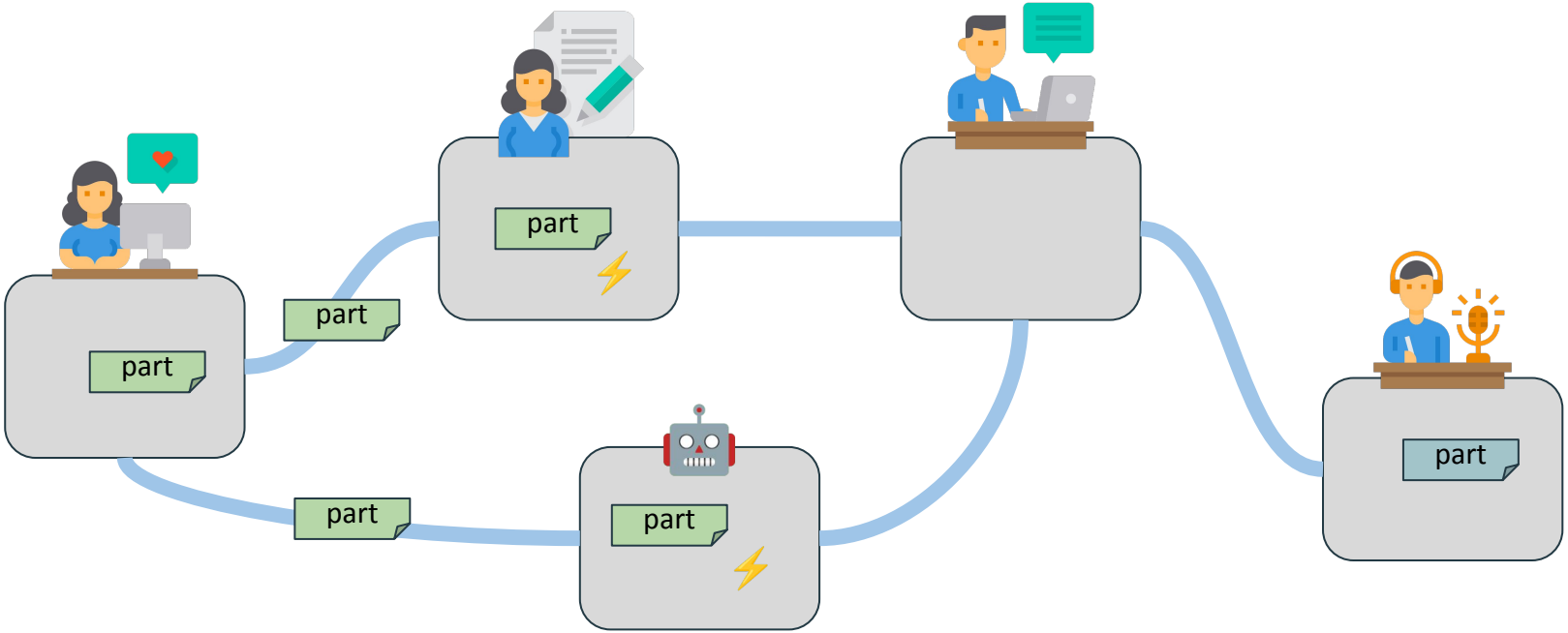


Challenge: Distributing Everything



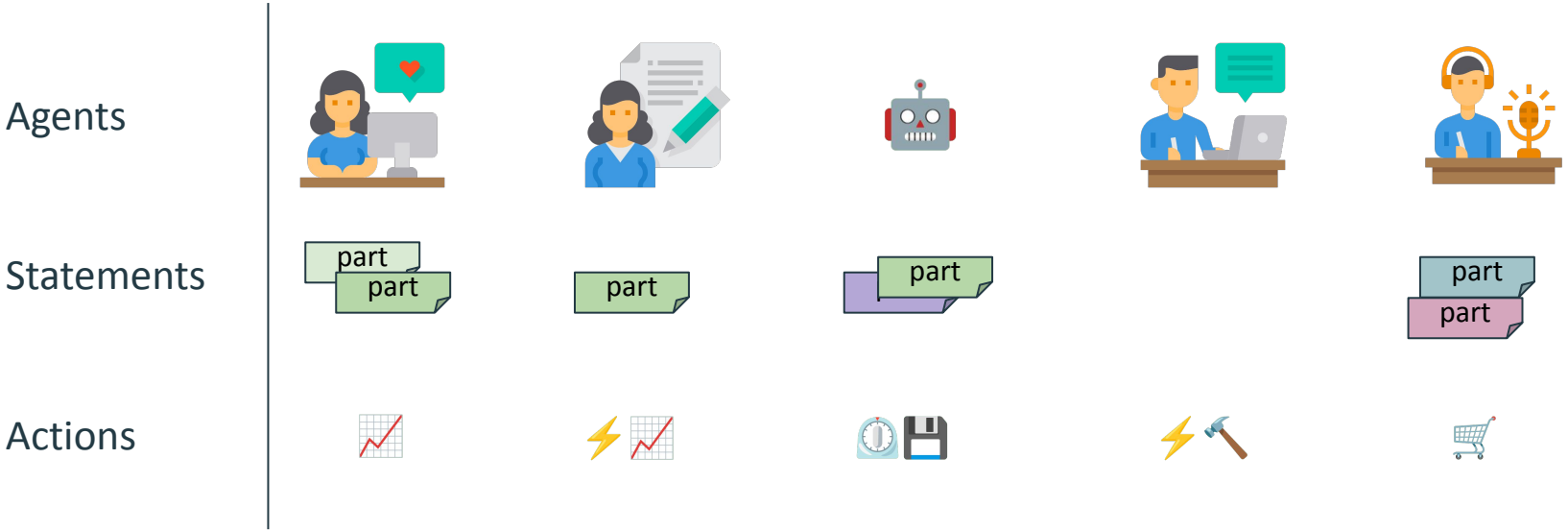
How can we map our vision onto the distributed system?

Approach: The JustAct Framework



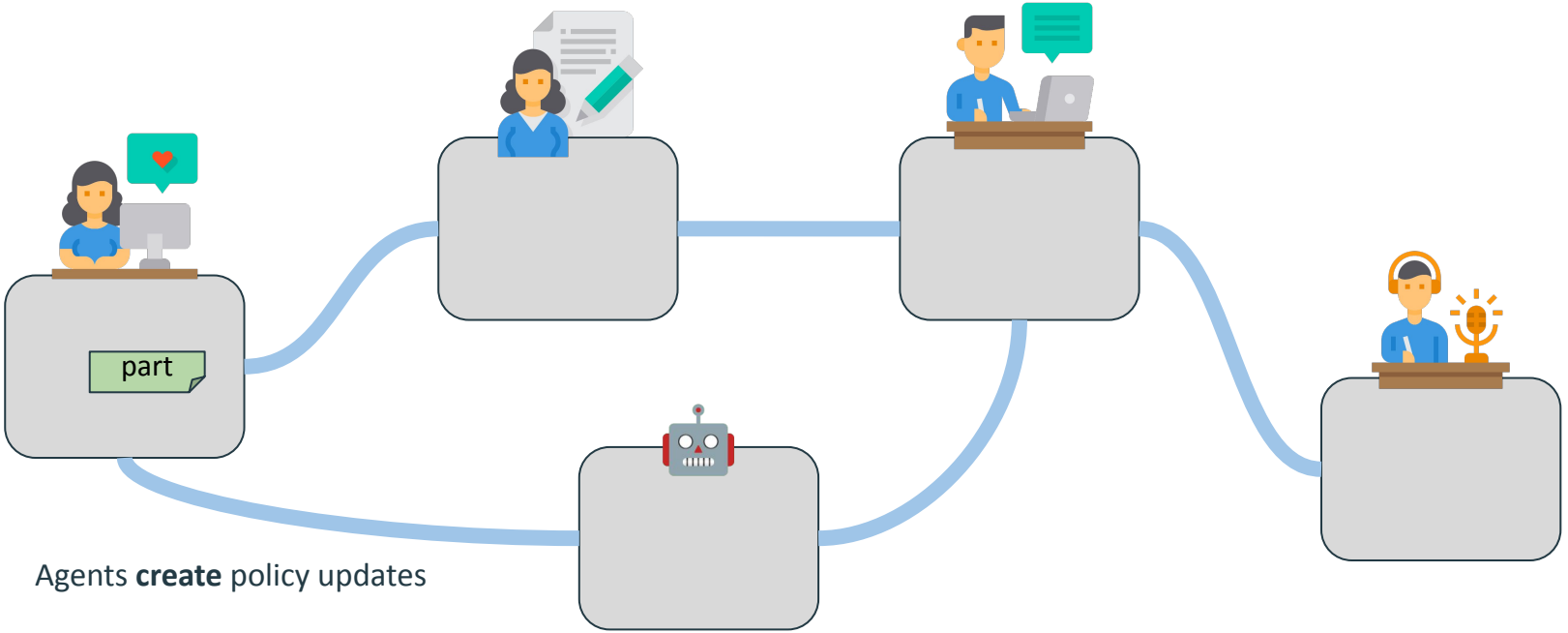
JustAct is designed around the distributed system

Approach: The JustAct Framework



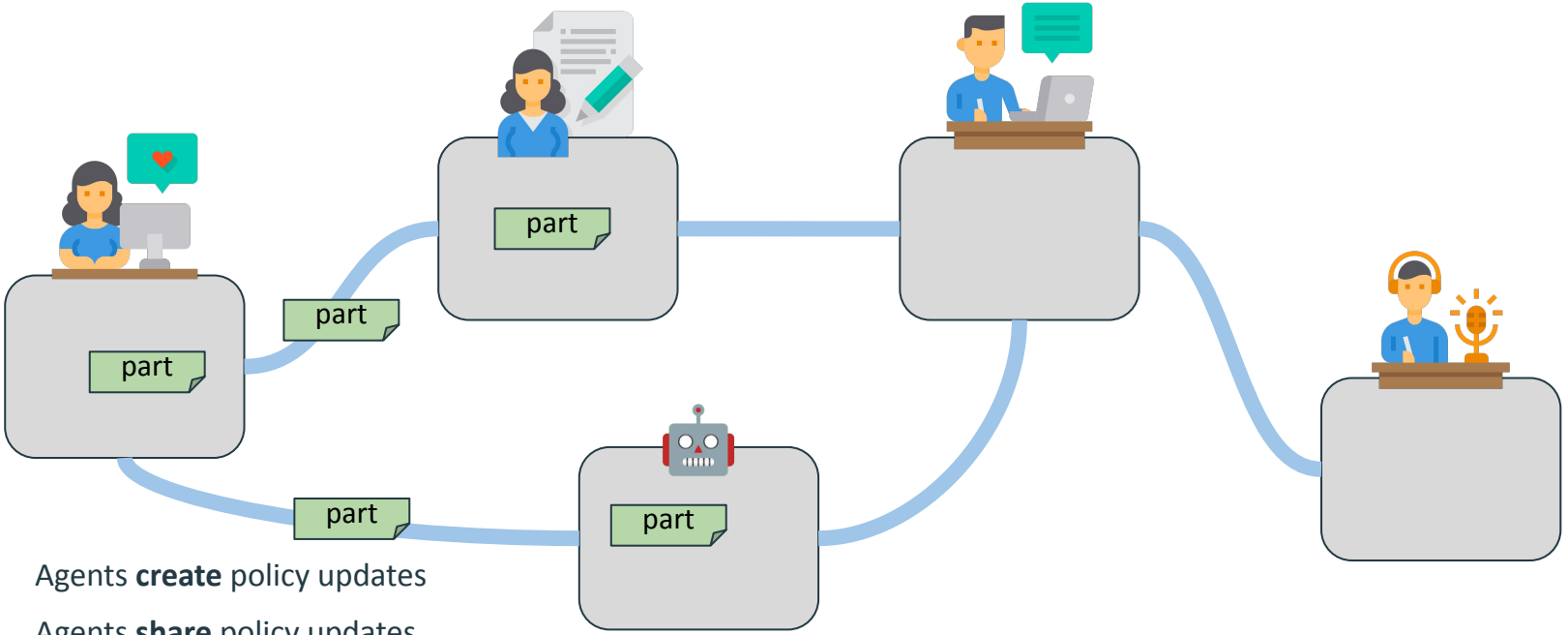
Policy and reasoning is distributed

Approach: The JustAct Framework



Agents **create** policy updates

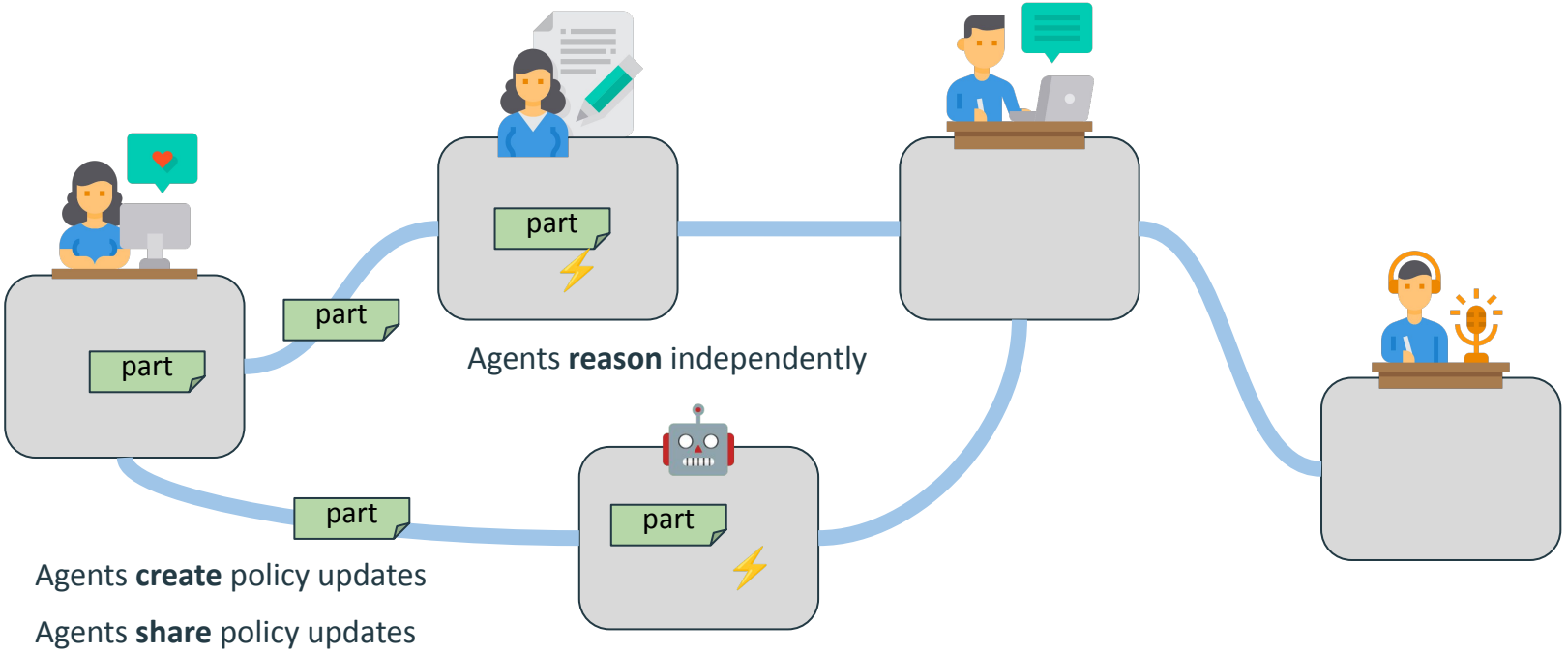
Approach: The JustAct Framework



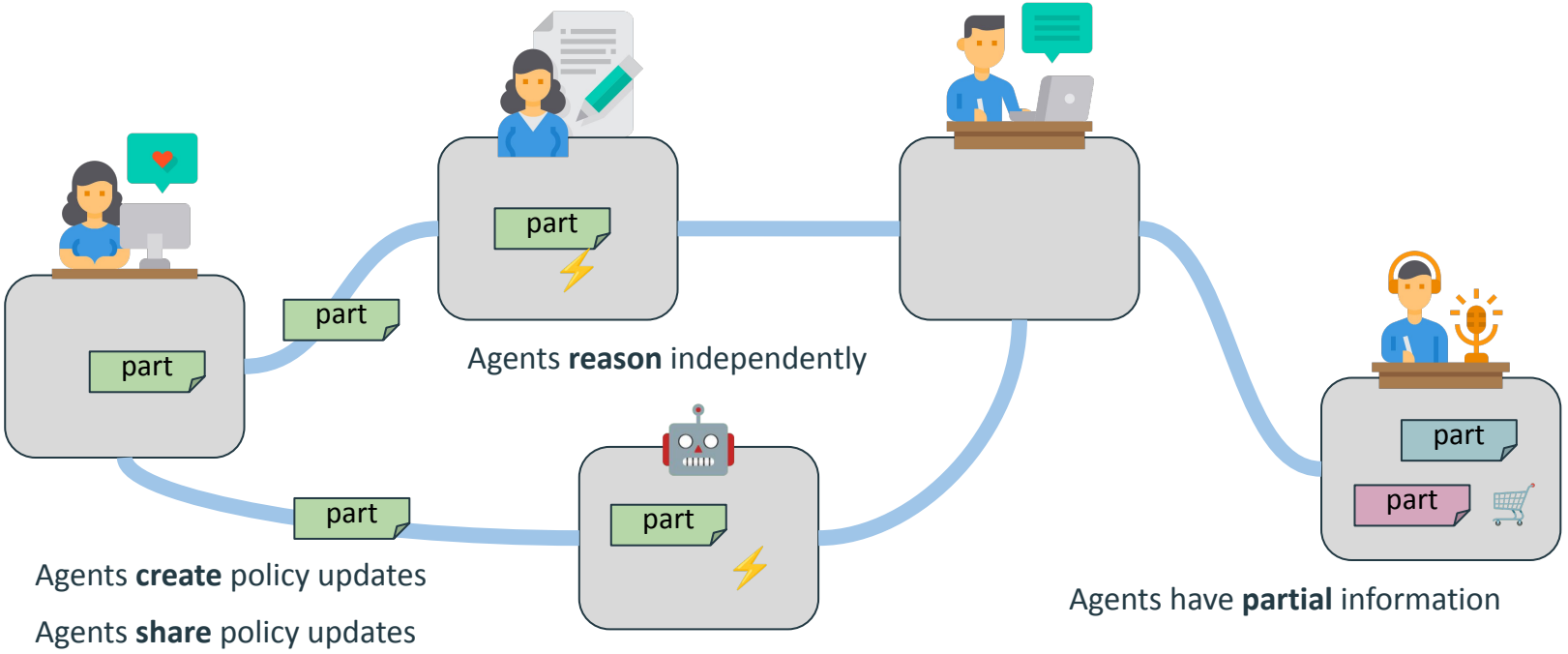
Agents **create** policy updates

Agents **share** policy updates

Approach: The JustAct Framework

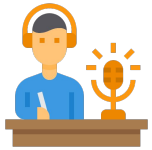


Approach: The JustAct Framework

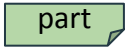


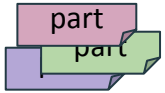
Approach: The JustAct Framework

Agents



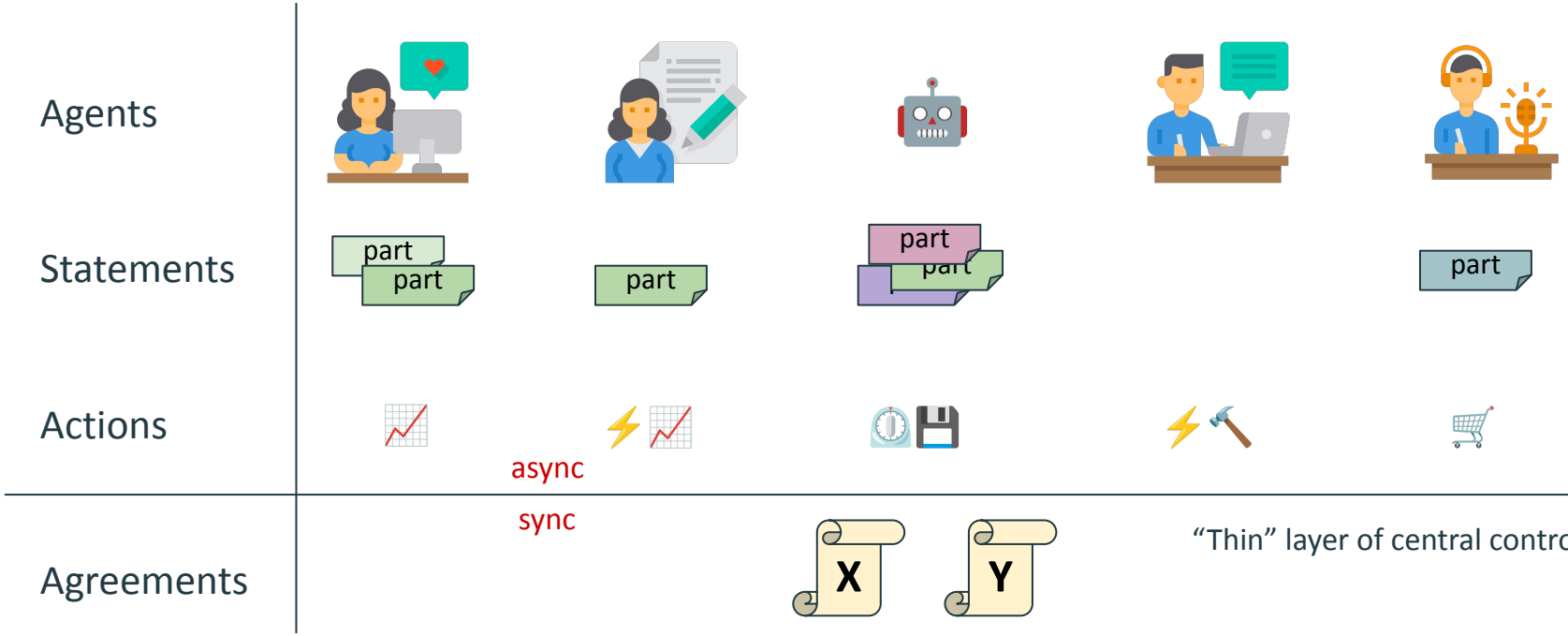
Statements



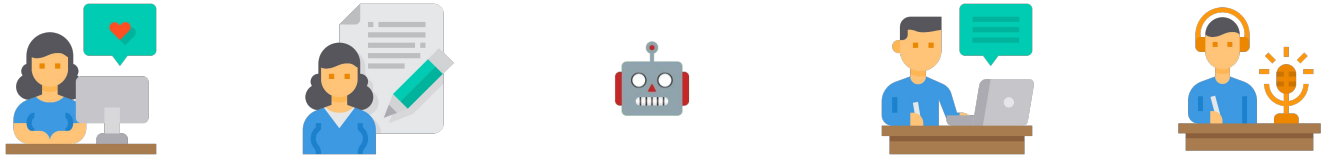


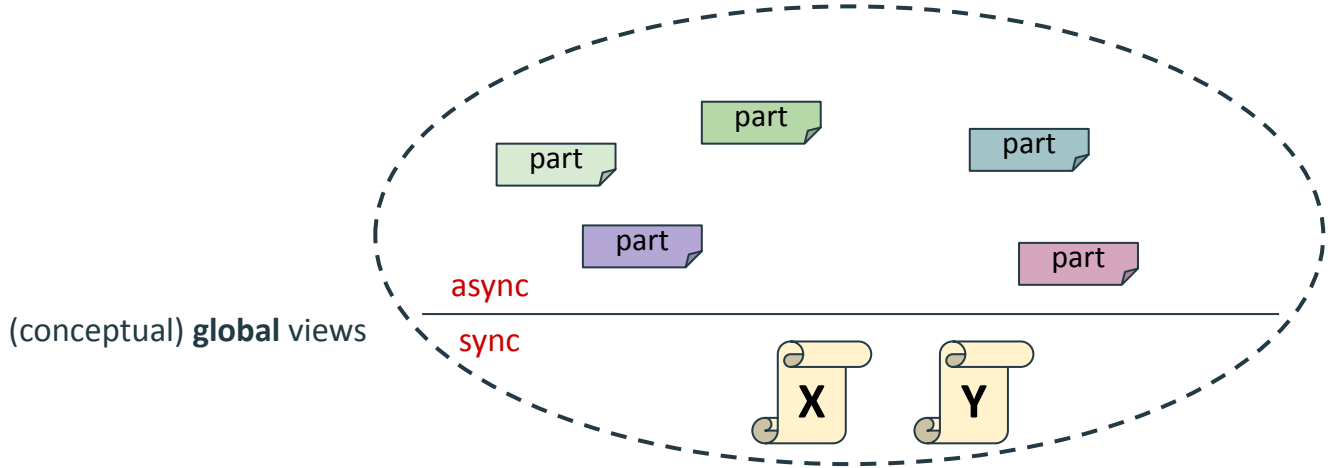
Actions

Approach: The JustAct Framework



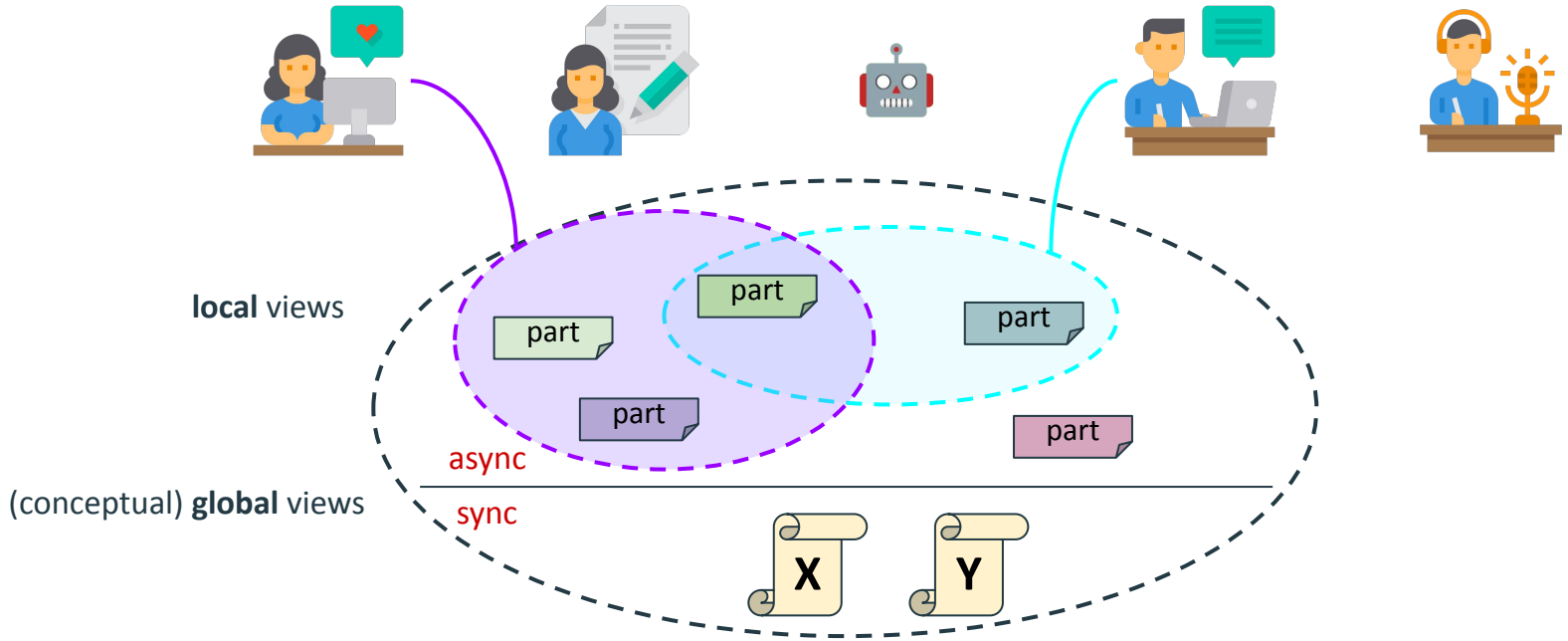
Approach: The JustAct Framework



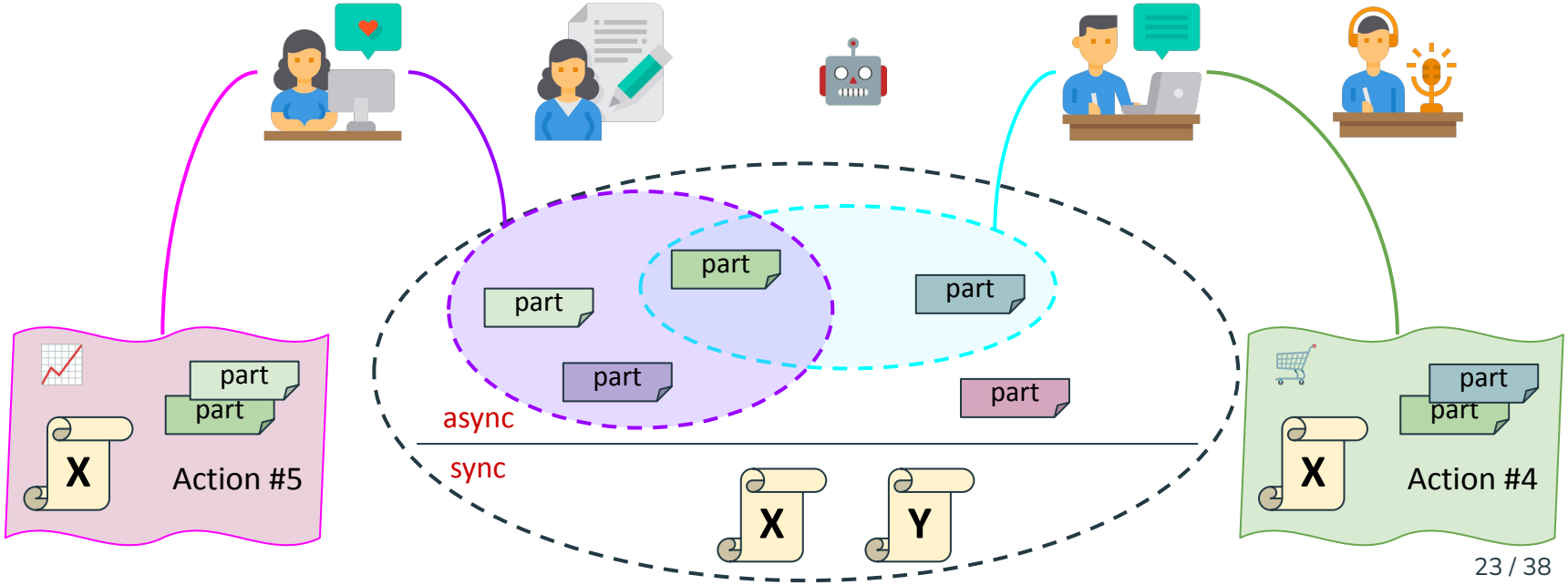


(conceptual) **global** views

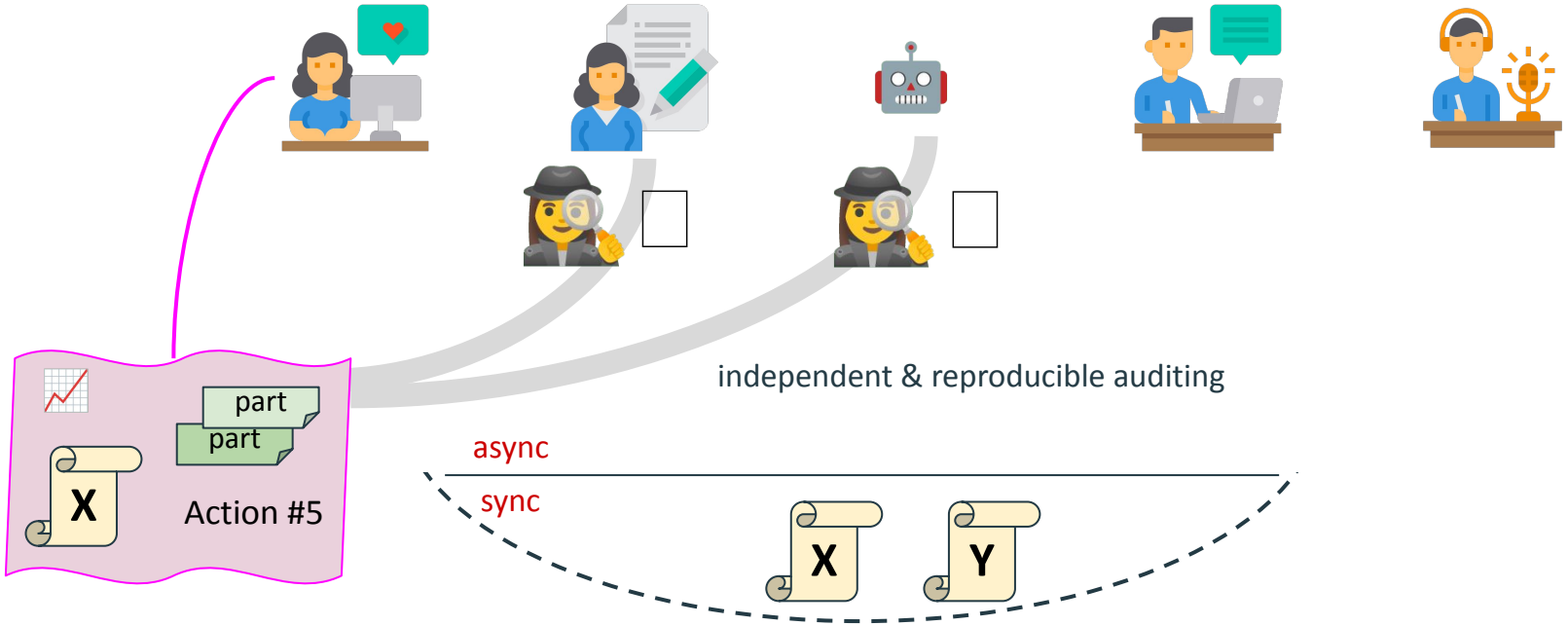
Approach: The JustAct Framework



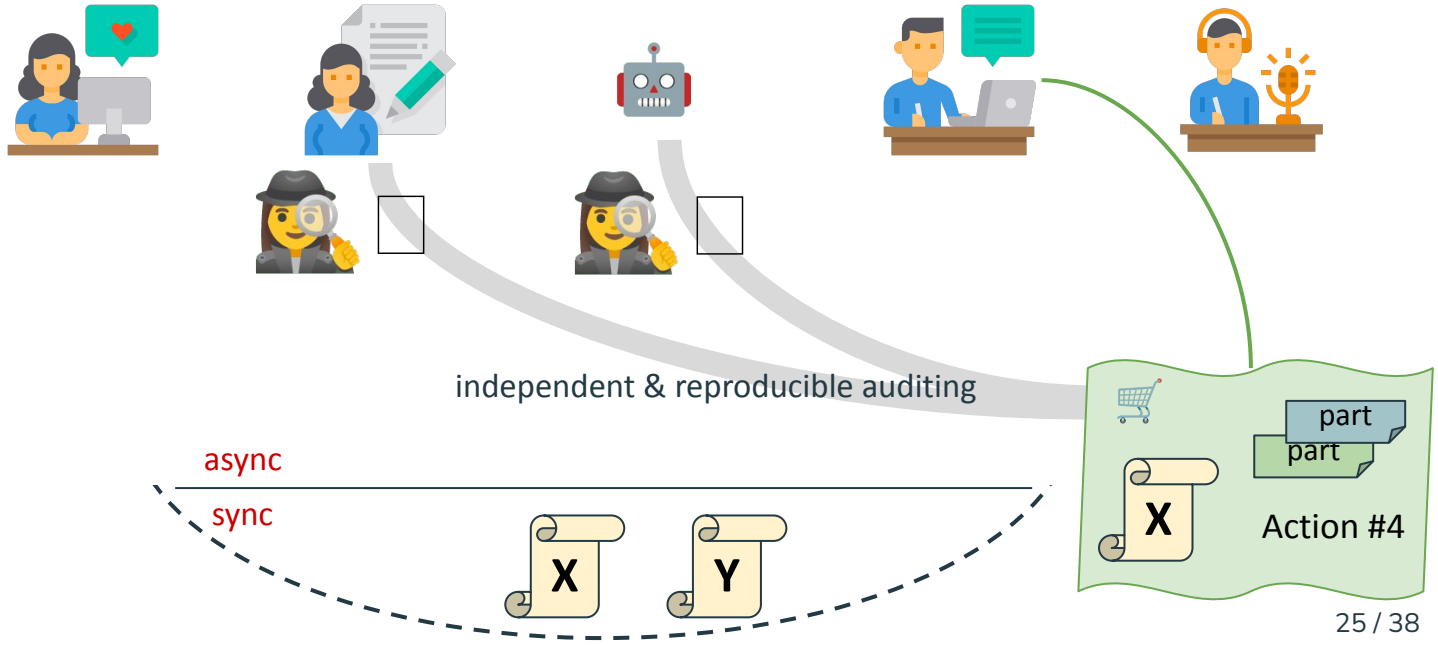
Approach: The JustAct Framework



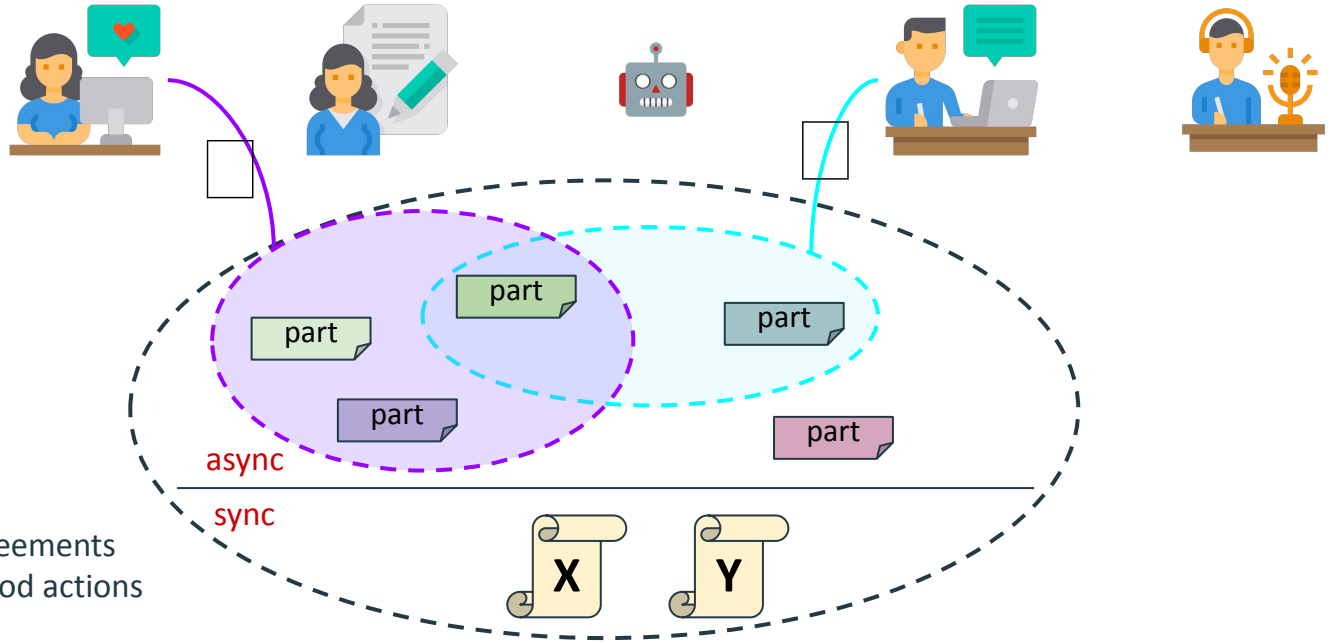
Approach: The JustAct Framework



Approach: The JustAct Framework

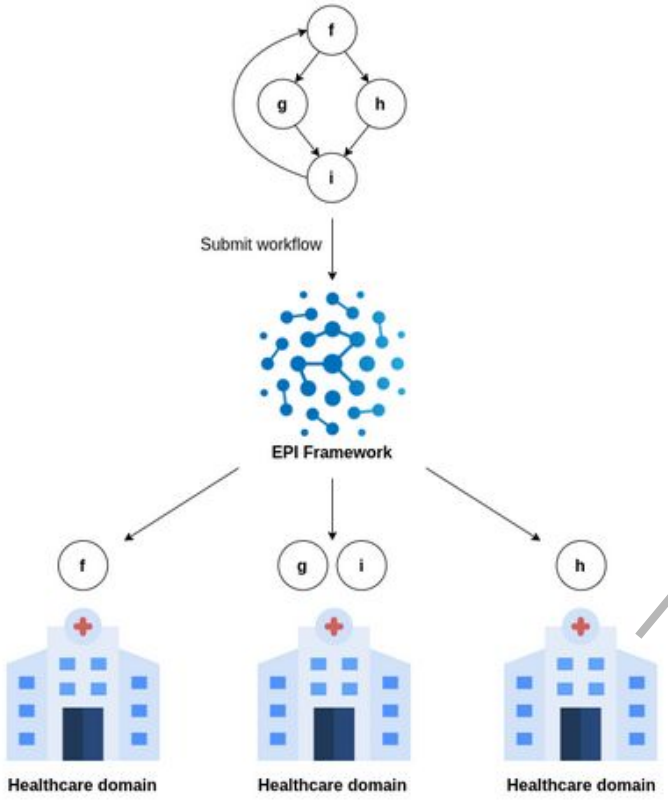


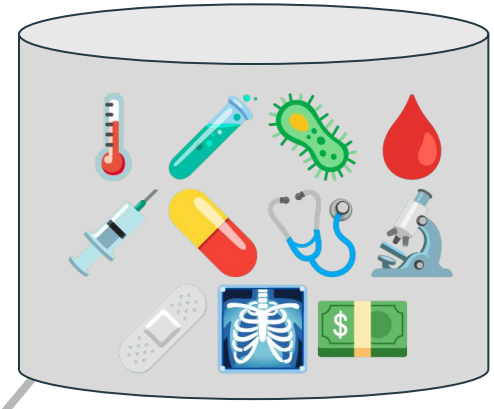
Approach: The JustAct Framework

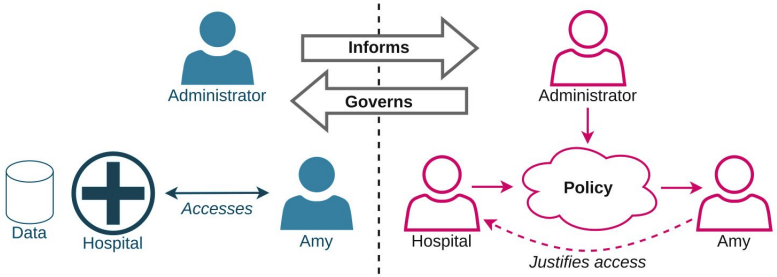


Agents choose agreements
that justify only good actions

Case Study


f
g
h
i
Submit workflow
EPI Framework
f
g
i
h
Healthcare domain
Healthcare domain
Healthcare domain

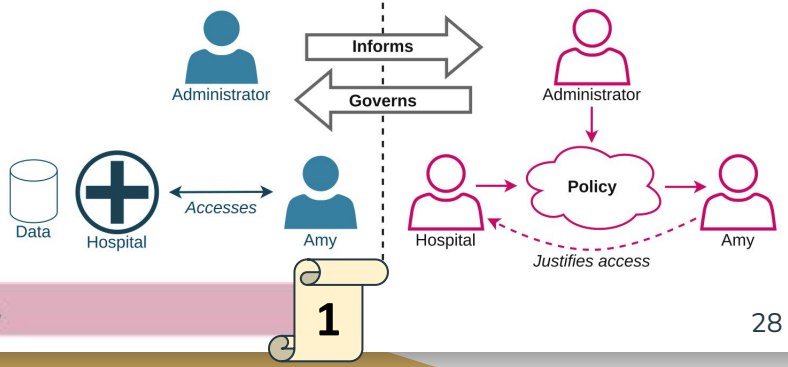



Administrator
Informs
Governs
Administrator
Data
Hospital
Accesses
Amy
Hospital
Policy
Amy
Justifies access

Case Study: Agreement 1

```
------------------ Part 1 ------------------
error if (Fact within Msg1) within Msg2.
error if (actor Agent)      within Msg2.

Sayer says Fact if Fact within (Sayer M) and diff { (consortium 1) (Sayer M) }.
error if Sayer says (Agent says Fact).
------------------ Part 2 ------------------
Sayer drives Task if Sayer says (Task ready).
Sayer drives Task if Sayer says (Task has input  Variable).
Sayer drives Task if Sayer says (Task has output Label).

error if Sayer says (Agent drives Task).
error if Agent drives (Driver Name) and diff { Agent Driver }.
------------------ Part 3 ------------------
error if Task executed and not Task ready.
error if Task2 executed and Task2 has input (Task1 Label) and not Task1 executed.
```

```
------------------ Part 4 ------------------
Worker reads  Variable     if Task has input Variable and actor Worker
                              and Worker says (Task executed).
Worker writes (Task Label) if Task has output Label   and actor Worker
                              and Worker says (Task executed).
error if Sayer says (Worker reads  Variable).
error if Sayer says (Worker writes Variable).
------------------ Part 5 ------------------
error if Worker says (Task executed) and (Task has input Variable) within Msg
    and Task involves Checker and not Checker says (authorise Task in Msg by Worker).

error if authorise Task in Msg1 by Worker
     and (Task has input Variable) within Msg2 and diff { Msg1 Msg2 }.
------------------ Part 6 ------------------
Task  involves Checker if Checker controls (Task Label).
Task2 involves Checker if Task2 has input (Task1 Label) and Task1 involves Checker.
```

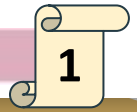


Agreement 1: Statement consortium 1 is agreed at time 1.

Case Study: Agreement 1

```
- - - - - - - - - - - - - - - - - - - - Part 4 - - - - - - - - - - - - - - - - - - - -
Worker reads  Variable     if Task has input Variable and actor Worker
                                and Worker says (Task executed).
Worker writes (Task Label) if Task has output Label   and actor Worker
                                and Worker says (Task executed).
error if Sayer says (Worker reads  Variable).
error if Sayer says (Worker writes Variable).
- - - - - - - - - - - - - - - - - - - - Part 5 - - - - - - - - - - - - - - - - - - - -
error if Worker says (Task executed) and (Task has input Variable) within Msg
    and Task involves Checker and not Checker says (authorise Task in Msg by Worker).

error if authorise Task in Msg1 by Worker
     and (Task has input Variable) within Msg2 and diff { Msg1 Msg2 }.
- - - - - - - - - - - - - - - - - - - - Part 6 - - - - - - - - - - - - - - - - - - - -
Task  involves Checker if Checker controls (Task Label).
Task2 involves Checker if Task2 has input (Task1 Label) and Task1 involves Checker.
```

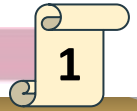
Agreement 1: Statement `consortium 1` **is agreed at time 1.**



Case Study: Defining Workflow Tasks

```
- - - - - - - - - - - - - - - - - - - - Part 4 - - - - - - - - - - - - - - - - - - - -
Worker reads  Variable     if Task has input Variable and actor Worker
                                and Worker says (Task executed).
Worker writes (Task Label) if Task has output Label   and actor Worker
                                and Worker says (Task executed).
error if Sayer says (Worker reads  Variable).
error if Sayer says (Worker writes Variable).
- - - - - - - - - - - - - - - - - - - - Part 5 - - - - - - - - - - - - - - - - - - - -
error if Worker says (Task executed) and (Task has input Variable) within Msg
    and Task involves Checker and not Checker says (authorise Task in Msg by Worker).

error if authorise Task in Msg1 by Worker
     and (Task has input Variable) within Msg2 and diff { Msg1 Msg2 }.
- - - - - - - - - - - - - - - - - - - - Part 6 - - - - - - - - - - - - - - - - - - - -
Task  involves Checker if Checker controls (Task Label).
Task2 involves Checker if Task2 has input (Task1 Label) and Task1 involves Checker.
```

Agreement 1: Statement `consortium 1` **is agreed at time 1.**



Case Study: A Workflow Task

Statement amy 1 (Amy prepares a patient-counting task).

```
(amy count-patients) has input ((surf        utils        ) entry-count).
(amy count-patients) has input ((st-antonius patients-2024) patients    ).
(amy count-patients) has output num-patients.
(amy count-patients) ready.
```

part

```
- - - - - - - - - - - - - - - - - - - - Part 4 - - - - - - - - - - - - - - - - - - - -
Worker reads  Variable     if Task has input Variable and actor Worker
                                and Worker says (Task executed).
Worker writes (Task Label) if Task has output Label   and actor Worker
                                and Worker says (Task executed).
error if Sayer says (Worker reads  Variable).
error if Sayer says (Worker writes Variable).
- - - - - - - - - - - - - - - - - - - - Part 5 - - - - - - - - - - - - - - - - - - - -
error if Worker says (Task executed) and (Task has input Variable) within Msg
    and Task involves Checker and not Checker says (authorise Task in Msg by Worker).

error if authorise Task in Msg1 by Worker
     and (Task has input Variable) within Msg2 and diff { Msg1 Msg2 }.
- - - - - - - - - - - - - - - - - - - - Part 6 - - - - - - - - - - - - - - - - - - - -
Task  involves Checker if Checker controls (Task Label).
Task2 involves Checker if Task2 has input (Task1 Label) and Task1 involves Checker.
```

Agreement 1: Statement consortium 1 **is agreed at time 1.**

1

Case Study: Authorised Task Execution

Statement amy 1 (Amy prepares a patient-counting task).

```
(amy count-patients) has input ((surf        utils        ) entry-count).
(amy count-patients) has input ((st-antonius patients-2024) patients    ).
(amy count-patients) has output num-patients.
(amy count-patients) ready.
```

part

Part 4

```
Worker reads  Variable     if Task has input Variable and actor Worker
                                and Worker says (Task executed).
Worker writes (Task Label) if Task has output Label   and actor Worker
                                and Worker says (Task executed).
error if Sayer says (Worker reads  Variable).
error if Sayer says (Worker writes Variable).
```

Part 5

```
error if Worker says (Task executed) and (Task has input Variable) within Msg
    and Task involves Checker and not Checker says (authorise Task in Msg by Worker).

error if authorise Task in Msg1 by Worker
     and (Task has input Variable) within Msg2 and diff { Msg1 Msg2 }.
```

Part 6

```
Task  involves Checker if Checker controls (Task Label).
Task2 involves Checker if Task2 has input (Task1 Label) and Task1 involves Checker.
```

Agreement 1: Statement consortium 1 **is agreed at time 1.**

1

Case Study: Conditional Authorisation

Statement amy 1 (Amy prepares a patient-counting task).

```
(amy count-patients) has input ((surf        utils        ) entry-count).
(amy count-patients) has input ((st-antonius patients-2024) patients    ).
(amy count-patients) has output num-patients.
(amy count-patients) ready.
```

part

Statement st-antonius 1 (St. Antonius asserts control of the patient data).

```
(st-antonius patients-2024) has output patients.
(st-antonius patients-2024) ready.
st-antonius controls ((st-antonius patients-2024) patients).

(st-antonius patients-2024) executed.
authorise (st-antonius patients-2024) in (st-antonius 1) by st-antonius
```

part

Part 4

```
Worker reads  Variable     if Task has input Variable and actor Worker
                                and Worker says (Task executed).
Worker writes (Task Label) if Task has output Label   and actor Worker
                                and Worker says (Task executed).
error if Sayer says (Worker reads  Variable).
error if Sayer says (Worker writes Variable).
```

Part 5

```
error if Worker says (Task executed) and (Task has input Variable) within Msg
    and Task involves Checker and not Checker says (authorise Task in Msg by Worker).

error if authorise Task in Msg1 by Worker
     and (Task has input Variable) within Msg2 and diff { Msg1 Msg2 }.
```

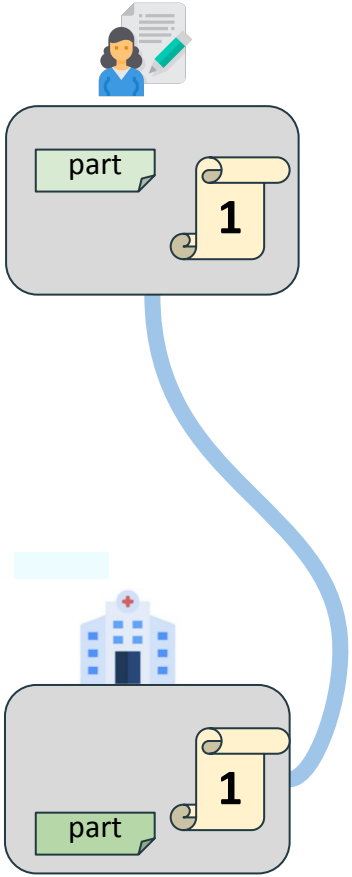
Part 6

```
Task  involves Checker if Checker controls (Task Label).
Task2 involves Checker if Task2 has input (Task1 Label) and Task1 involves Checker.
```

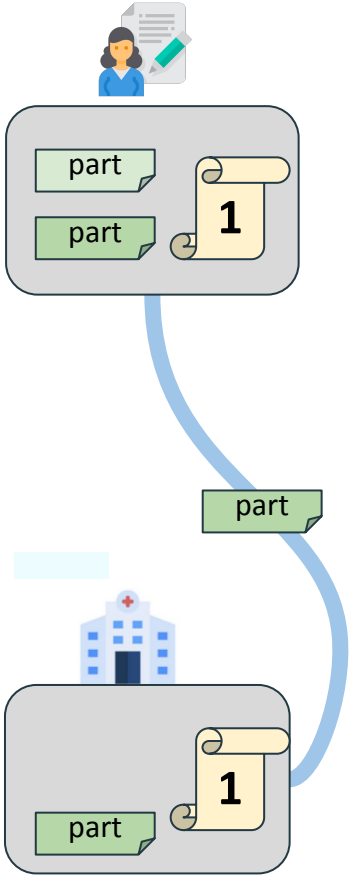
Agreement 1: Statement consortium 1 is agreed at time 1.

1

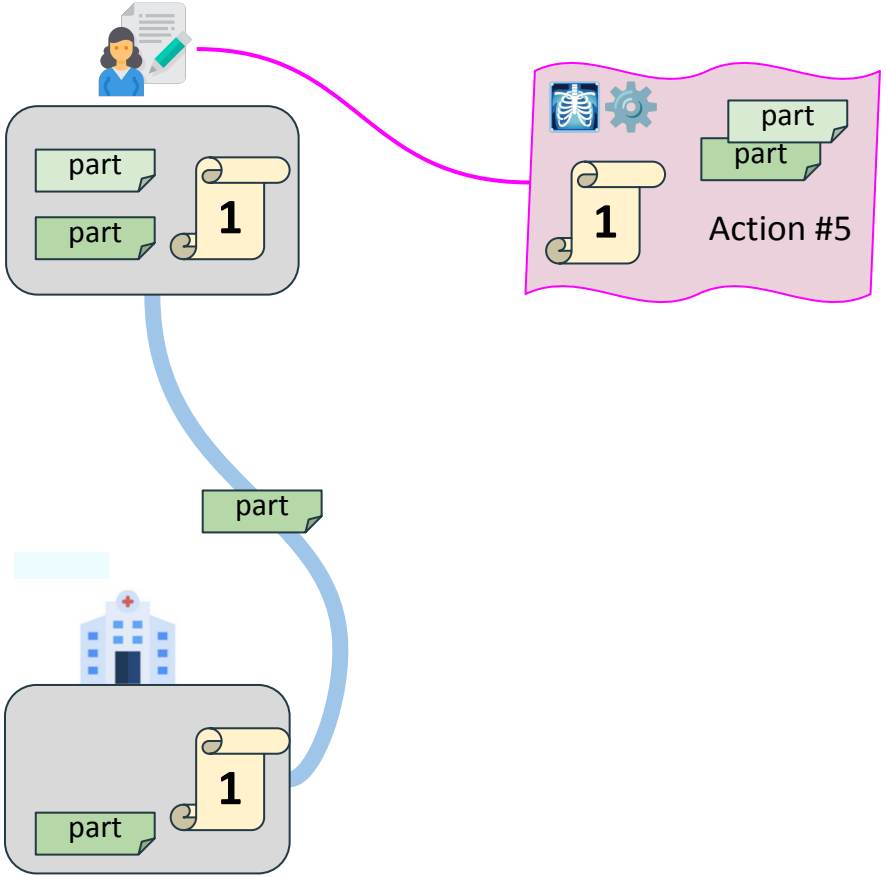
Case Study: Partial Views



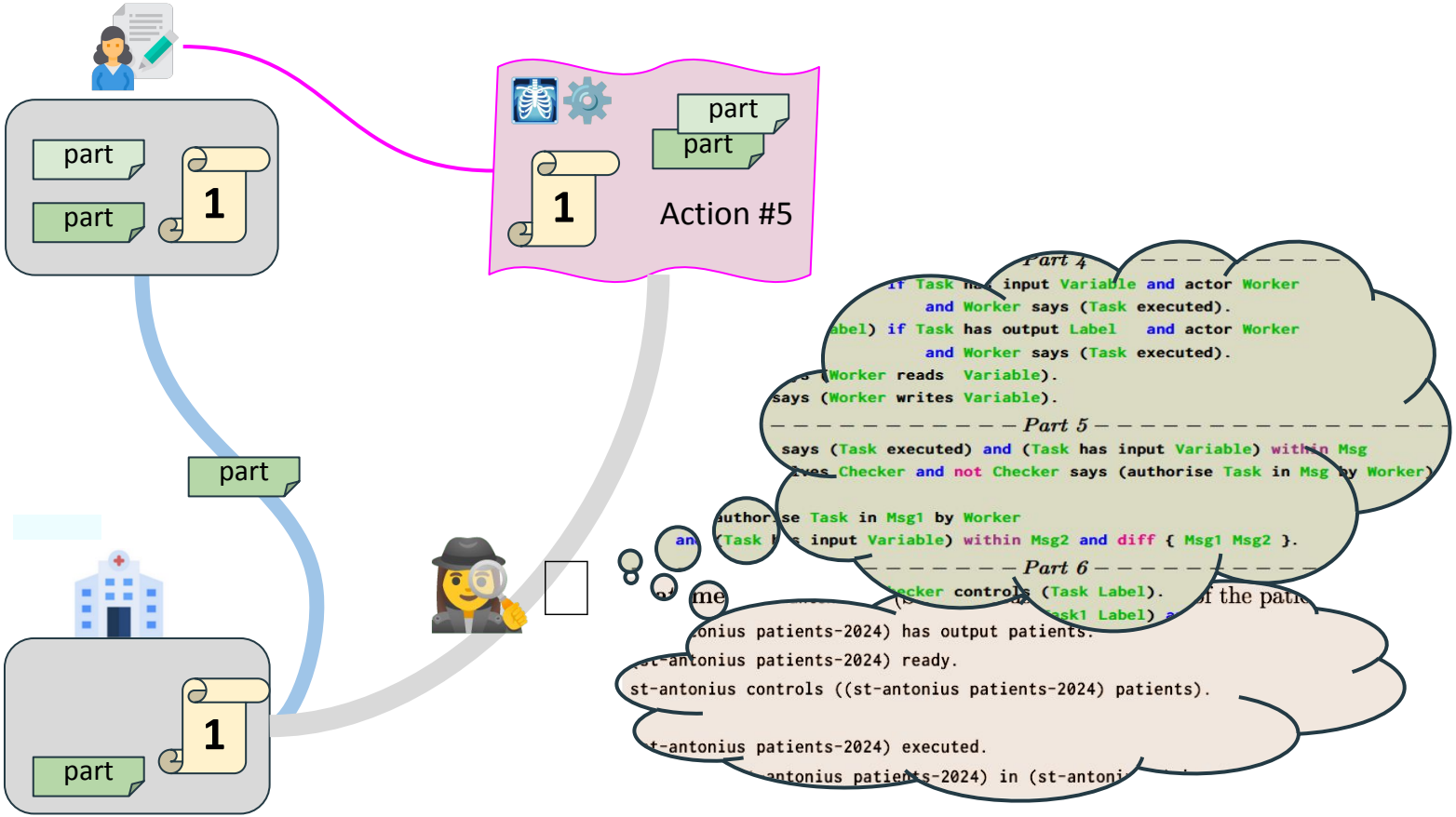
Case Study: Communication



Case Study: Justified Action



Case Study: Auditing



Bonus: Where to Find the Details

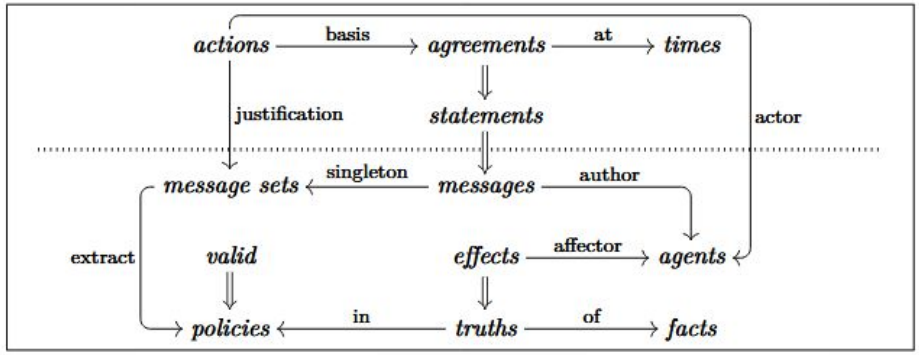


Figure 2.2: Graphical depiction of the *framework ontology*. Sets (italicized) are related by pure, total functions (→ and ⇒ arrows) from domain to co-domain. Each (⇒) denotes an identity function, *e.g.*, $statements \subseteq messages$. Functions are identified by their co-domain (or by a label if given). The dotted line distinguishes sets and functions that are *dynamic* (above) and *static* (below). At runtime, new elements may be added to dynamics, but statics are fixed.

Definition 2.2. $well\text{-}behaved(\alpha : agents) \triangleq$
$$\forall a : actions, \quad (actor(a) = \alpha) \rightarrow permitted(a).$$

Definition 2.3. $permitted(a : actions) \triangleq$

$$\begin{aligned} & justification(a) \subseteq statements && \text{(stated justification)} \\ \wedge\ & agreed(basis(a)) \in justification(a) && \text{(based justification)} \\ \wedge\ & extract(justification(a)) \in valid && \text{(valid justification)} \\ \wedge\ & at(basis(a)) \in current. && \text{(current action)} \end{aligned}$$

Definition 2.5 (Message Identifiers). $messageIds \triangleq facts \times facts.$

Definition 2.6. $messages \triangleq messageIds \times policies.$

Definition 2.7. $payload(\langle i, p \rangle : messages) \triangleq p.$

Definition 2.8. $author(\langle i, p \rangle : messages) \triangleq author'(i)$
where $author'(\langle f, f' \rangle : messageIds) \triangleq f.$

(published) *JustAct: Actions Universally Justified by Partial Dynamic Policies* https://tinyurl.com/justact1

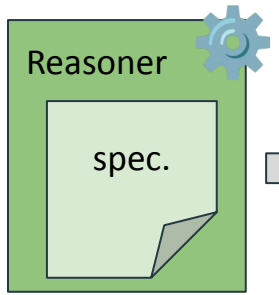
(extended) *JustAct+: Justified and Accountable Actions in Policy-Regulated, Multi-Domain Data Processing* https://tinyurl.com/justact2

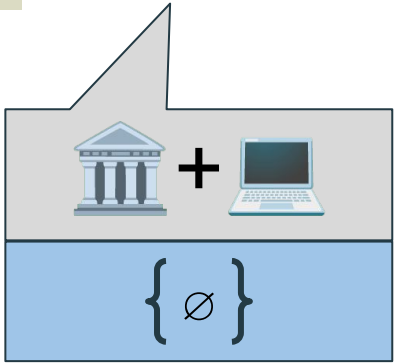
End Collage

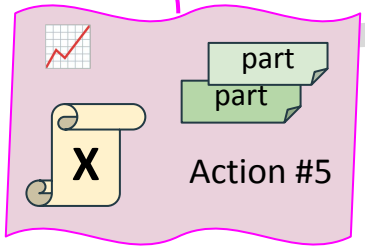
Contact:
c.a.esterhuyse@uva.nl

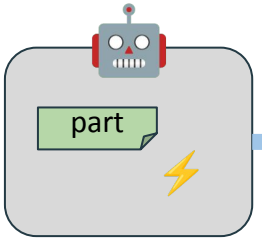
Statement amy 1 (Amy prepares a patient-counting task).

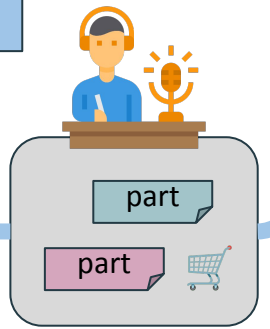
```
(amy count-patients) has input ((surf        util          ) entry-count).
(amy count-patients) has input ((st-antonius pa  ents-2024) patients   ).
(amy count-patients) has output num-patients.
(amy count-patients) ready.
```



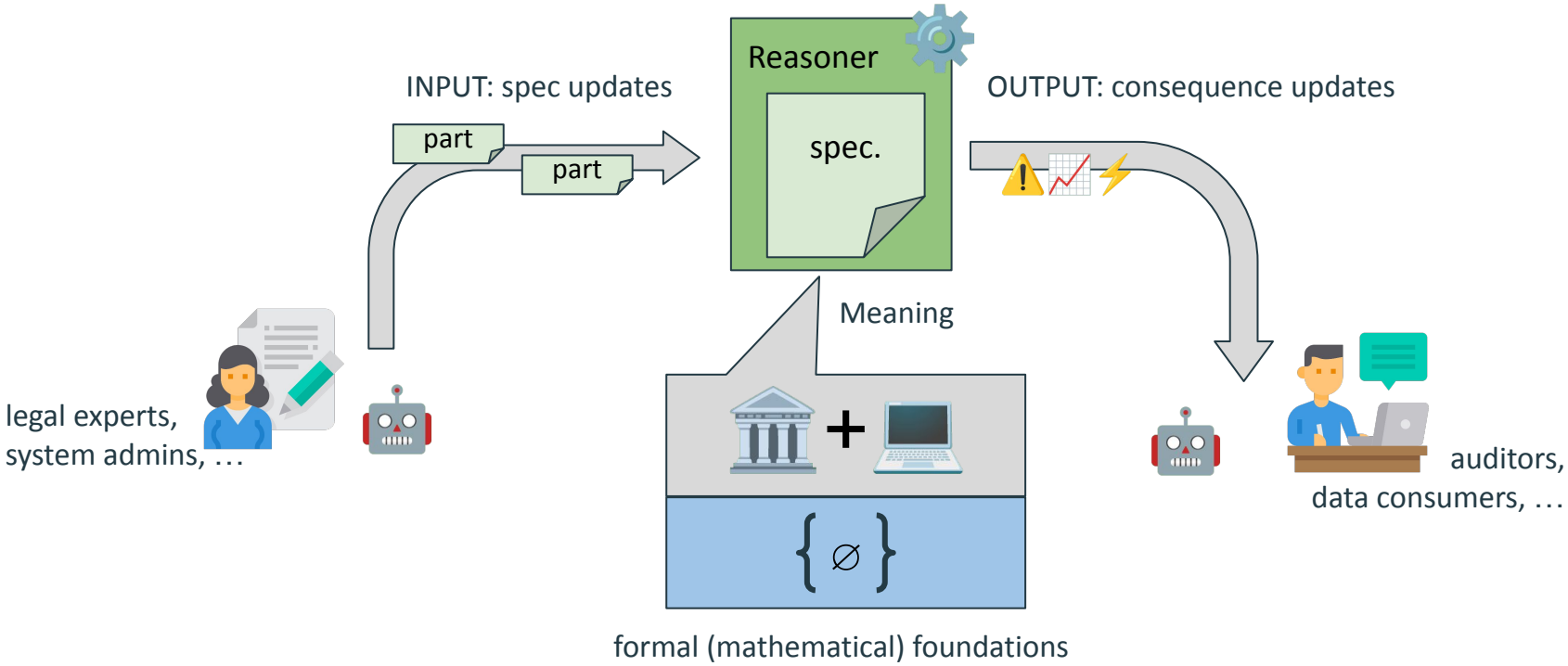




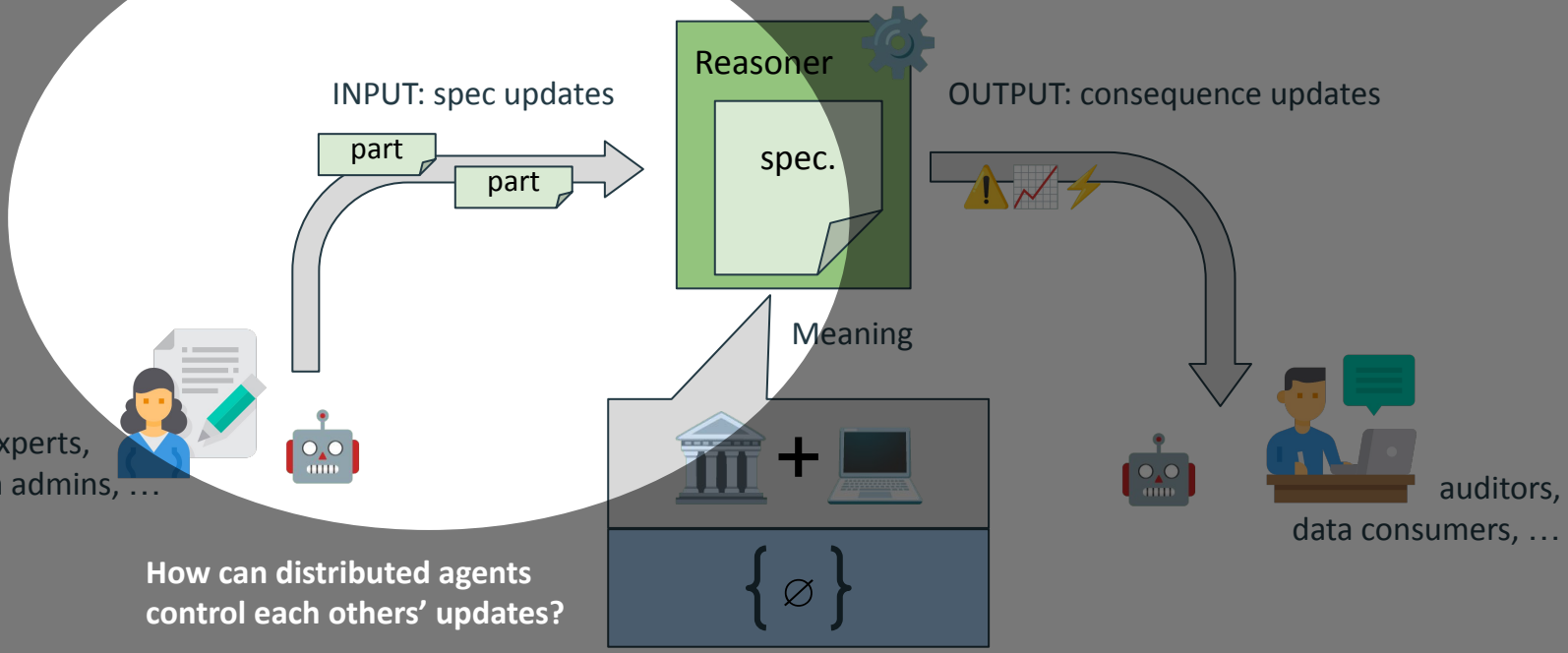




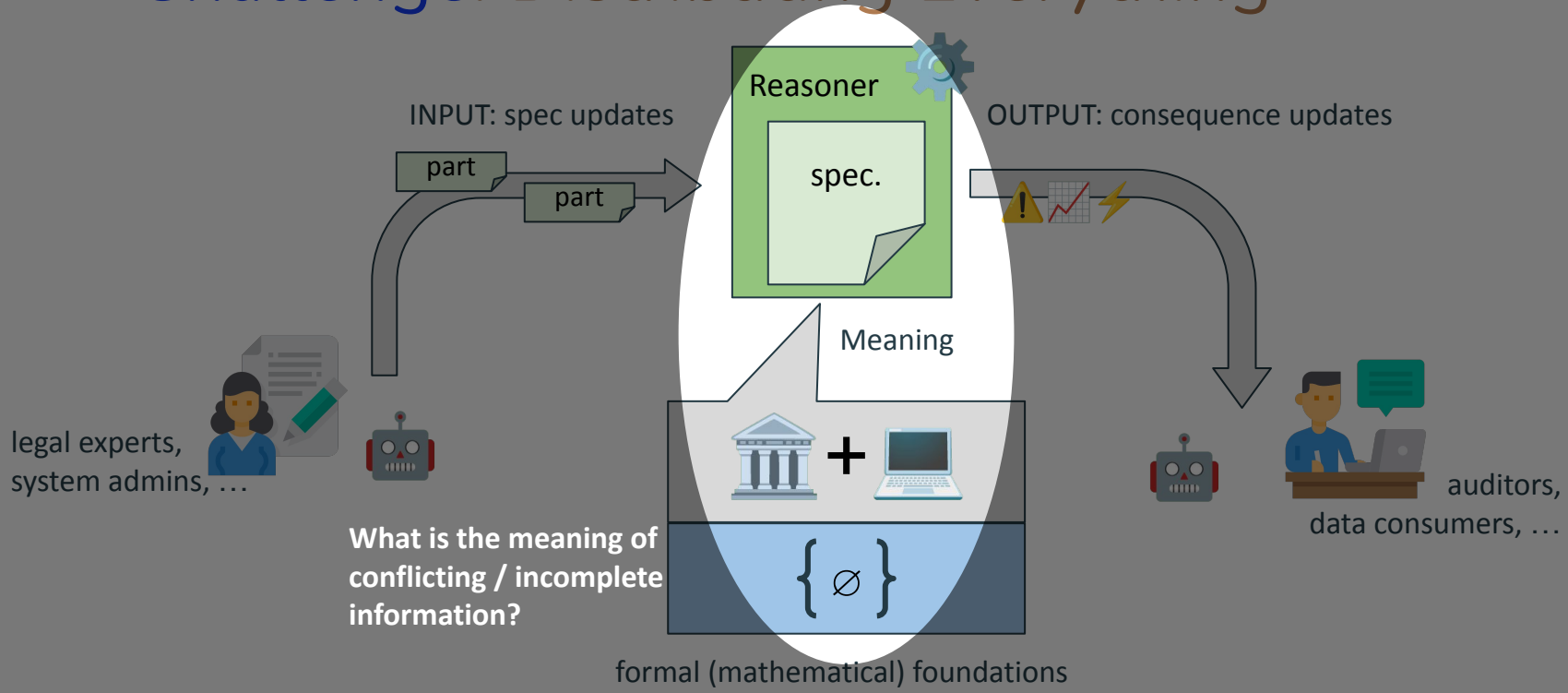
Challenge: Distributing Everything



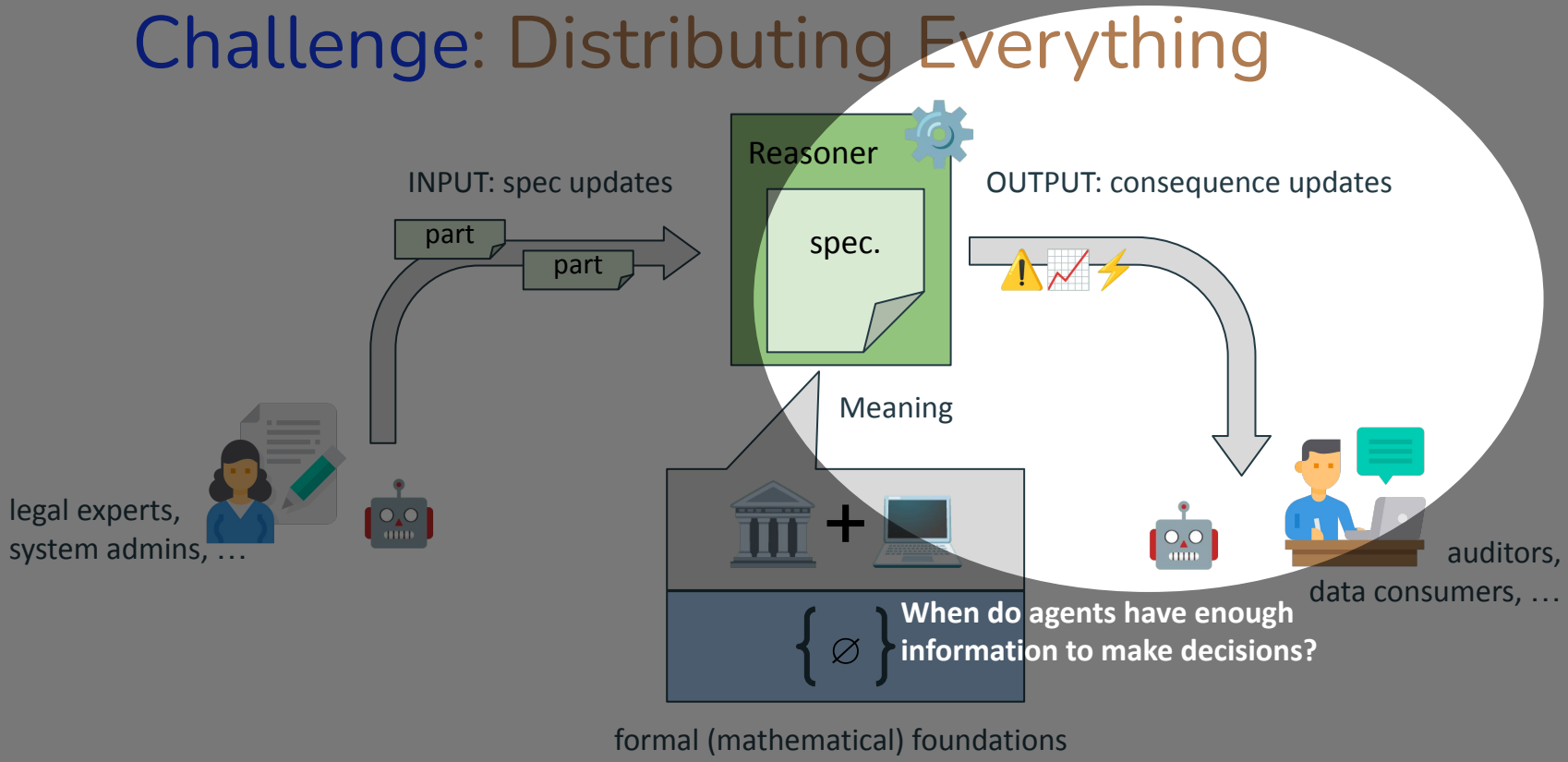
Challenge: Distributing Everything



Challenge: Distributing Everything



Challenge: Distributing Everything



University of Amsterdam
Institute for Information Law

UvA Research Update

Towards a generic legal data governance framework for B2G3P

Dr. H.L. Janssen | DMI | Research Update | 21 May 25

Introduction

Dr. Heleen L. Janssen

Assistant professor

Institute for Information Law (2020)

Projects

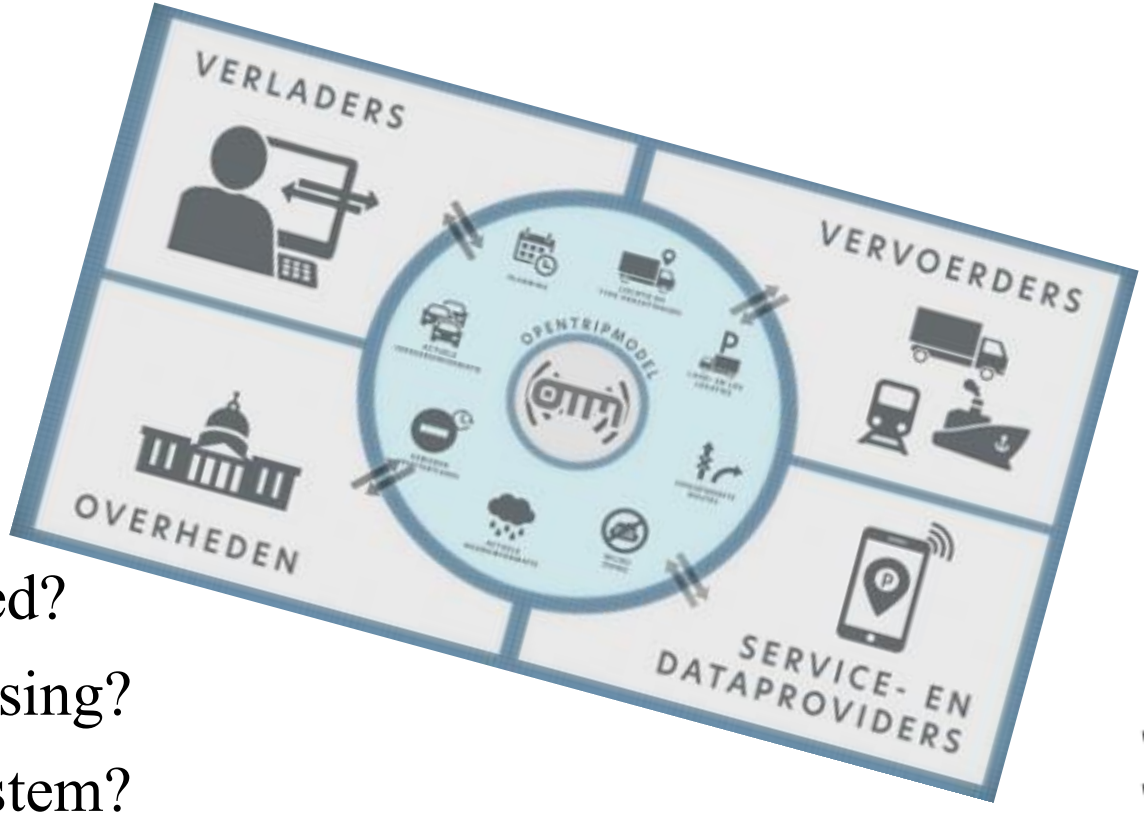
- 'Towards a generic legal data governance framework for B2G3P' (2024 –)
- 'From policy to practice in data governance and responsible data stewardship' (2024)
- 'Legal framework for local B2G (2023)

B2G3P data sharing is about…

Setting the legal scene for B2G3P

WHO has the power to process data?

Accountability and responsibility



- Who is stakeholder?
- Type entity – private or public?
- Type of data (and pertaining interest) involved?
- Who takes decisions over data and its processing?
- Roles present in the data intermediary ecosystem?
- What happens *in fact* in the data sharing ecosystem?

Choices are shaped by legal framework and by technical architecture

Legal framework for local B2G

- Research: mapping the legal landscape (2023)
  - Legal doctrinal & empirical research (stakeholder & expert workshop & employee interviews)
  - See https://www.ivir.nl/publicaties/download/Gemeentelijke_grip_op_private_sensorgegevens.pdf

- Purpose of B2G: improving execution of public tasks, more accurate policymaking
  - Access to data held by businesses
  - Mapping of applicable legal framework
  - Legal gaps, constraints and pitfalls

B2G mapping applicable legal framework

EU legal framework regulating B2G (horizontal laws)

- Fundamental rights
- GDPR, e-Privacy, Free flow of non-personal data
- IP law, Trade secrets, Database law,
- Data Act, Data Governance Act, AI Act
- Freedom of Information Act, Open Data Directive, Reuse of Public Sector Data

National legal instruments regulating B2G – General Administrative Law Act

- Public law: permit, regulation, subsidy
- Private law: ordinary contract, public procurement contract, concession

B2G – challenges

- EU law: procedural obligations & open norms
- Patchwork: uncertainty over legal coherence and consistence
- B2G often *ad hoc,* conditions unilaterally determined by businesses
- Refusal by businesses: legal and political-economic reasons
- What's in it for businesses?
- Exact problem definition is complex
- More legal obligations might not necessarily help B2G
- Citizen's interests?

➢ Research whether an independent, third party can help overcome some issues

B2G3P - Data sharing via a third party

"Data Intermediary" – key features:

- independent entity
- charged with effecting a governance regime around supplier and recipient rights and interests
- guides, constrains and monitors data use to ensure compliance
- can (re)distribute control over data and processing
- shape and (re)balance relationships between stakeholders
- ability to address data access and sharing challenges

Takeaways

- EU law provides basic guidelines, not a conclusive framework
- Legal uncertainty over broader applicable legal framework
- Generic legal framework: guide for data intermediary ecosystem stakeholders to apply the law
- Find sweet spot between generic and concrete applicability of the data governance framework

Questions & discussion