Rules as Executable Code for Adaptable and Accountable Software

L. Thomas van Binsbergen

Informatics Institute, University of Amsterdam
l.t.vanbinsbergen@uva.nl

March 10, 2026

Introduction

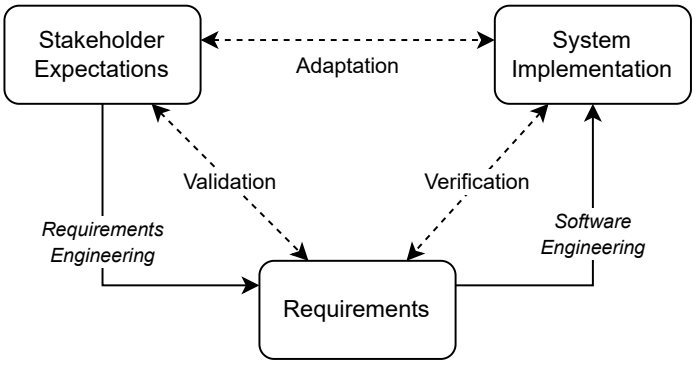


Figure: A depiction of the 'conventional' software engineering process.

Introduction

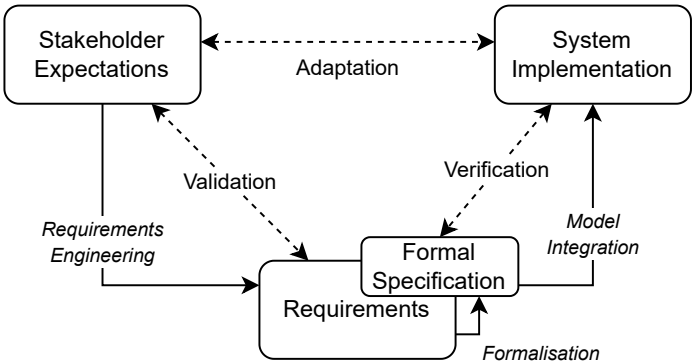


Figure: A depiction of a software engineering process involving formal verification and model-driven engineering.

Introduction

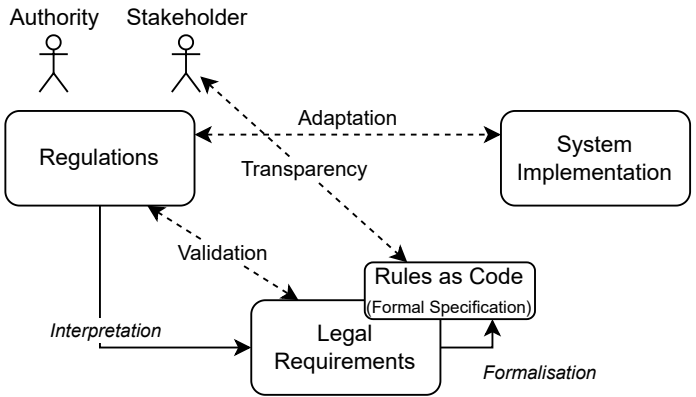


Figure: The legal slice of the formal verification and model-driven engineering process.

Introduction

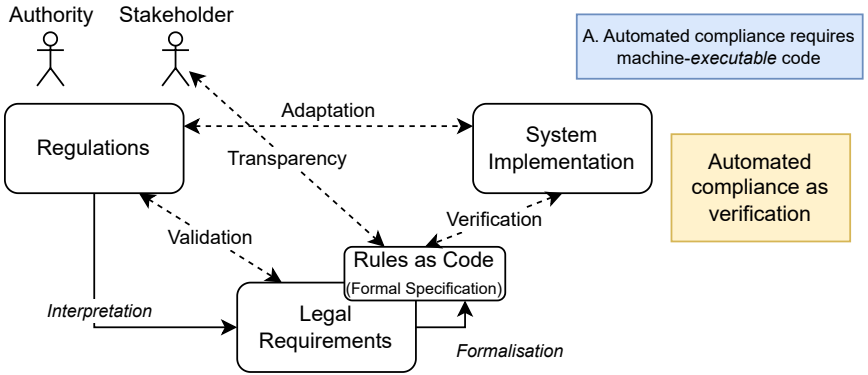


Figure: The legal slice of the formal verification and model-driven engineering process.

Introduction

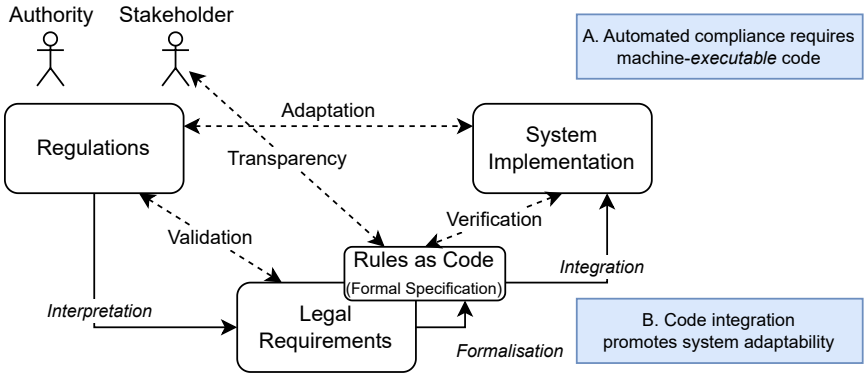


Figure: The legal slice of the formal verification and model-driven engineering process.

Introduction

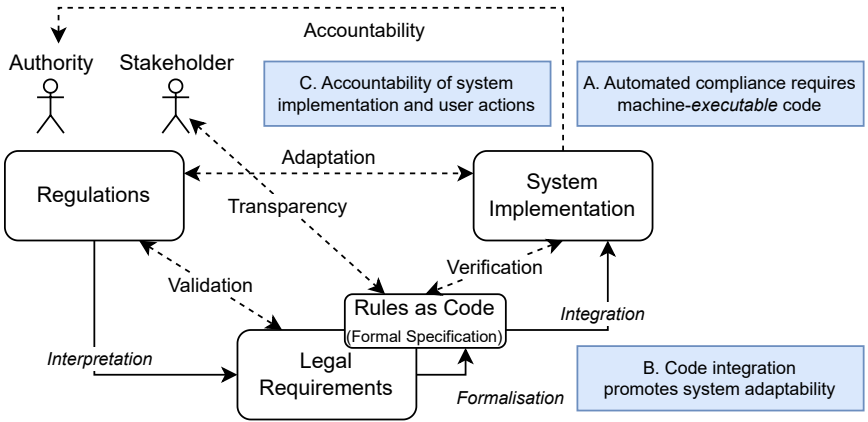


Figure: The legal slice of the formal verification and model-driven engineering process.

Introduction

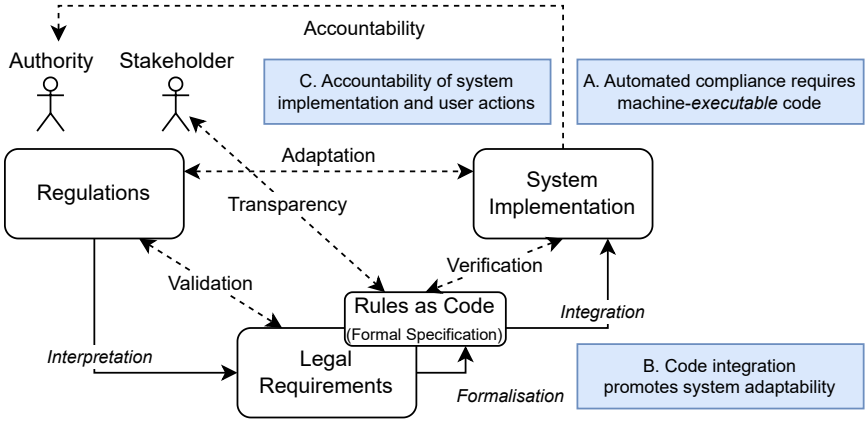
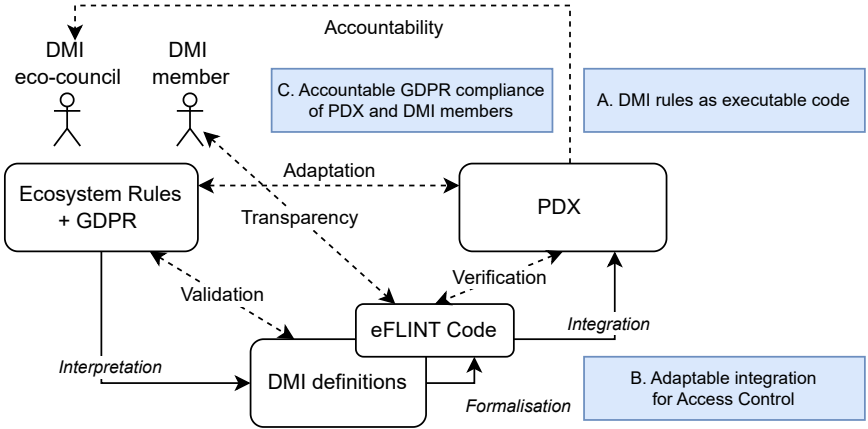


Figure: The legal slice of the formal verification and model-driven engineering process.

Overarching Research Question: What language semantics supports A through C?

The Example Case – DMI & PDX



- DMI definitions: `https://definities.dmi-ecosysteem.nl/`
- eFLINT Code: `https://gitlab.com/eflint/formalisations/dmi-afsprakenstelsel`

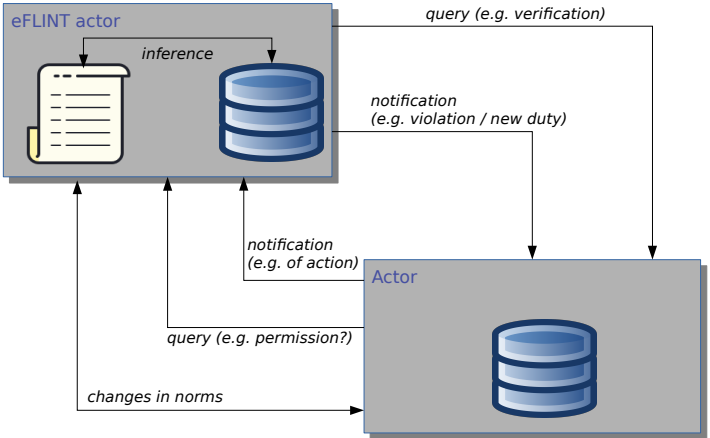
Overview

1. A. Machine-executable legal code
2. B. Integration: Adaptability through Separation of Concerns
3. C. Accountability of System and User Actions
4. Reflections

A. Machine-executable legal code

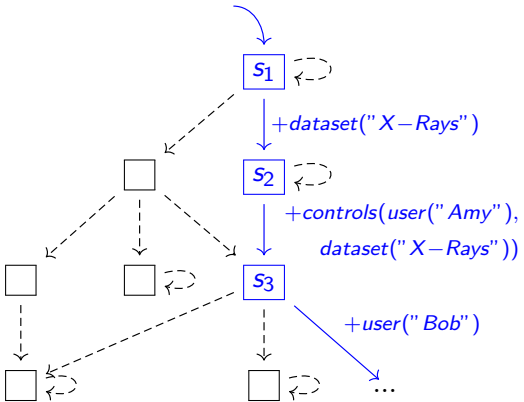
Normative Specification Language eFLINT

- Domain-specific language coupling normative/legal to computational concepts
- Based on logic programming and inference to derive a knowledge base
- Based on imperative programming to model state transitions



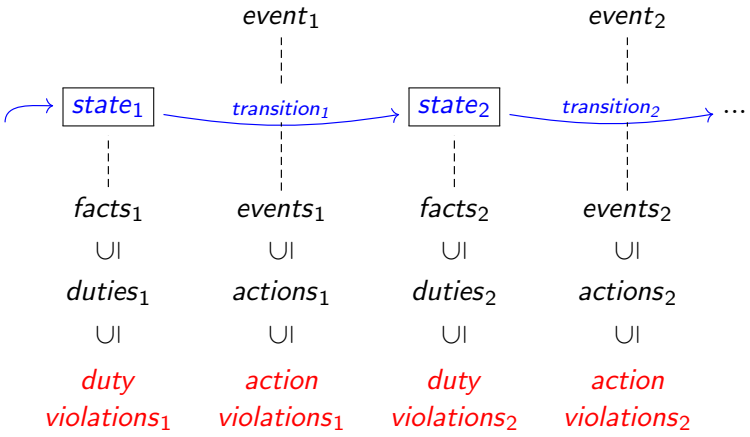
Semantics of Specifications and Scenarios

Specifications define a state machine:
i.e., which states exist, how they transition, and which are considered violating.
Scenarios trace a path within the state machine.



Checking for Violations

Per specification, the **input** is a scenario, and
the **output** is all violations and (a subset of) the facts and triggered events.



Example – DMI membership rules

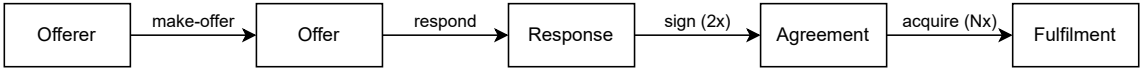
```
Fact date      Identified by Int  // latvian format YYYYMMDD
Fact member    Identified by String
Fact ready     Identified by member * date

Fact membership-application Identified by member * date
Fact membership-fee-payment Identified by member * date
Fact membership-approved    Identified by member * date

Fact membership-withdrawn Identified by member * date

Extend Fact ready Holds when
       membership-fee-payment(member, date') && date >= date'
    && membership-approved(member, date') && date >= date'
    && Not(membership-withdrawn(member, date')) && date' >= date
```

Example – DMI Transaction Process (1)



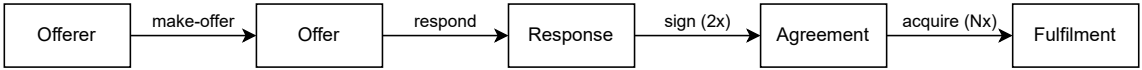

```
Placeholder offerer      For member
Placeholder respondent   For member

Fact dossier     Identified by Int
Fact offer       Identified by offerer * pdd * dossier
Fact response    Identified by respondent * offer
Fact pdd         Identified by String // PDD: PDX product or service

Act make-offer Actor offerer
  Related to pdd, date, dossier
  Holds when offerer // make-offer is available in states with offerers
  Creates offer() // dossier, offerer and pdd arguments are implicit

Act respond Actor respondent
  Related to offer, date
  Holds when offer // respond is available in states with offers
  Creates response()
```

Example – DMI Transaction Process (2)




```
Fact agreement  Identified by response
Fact fulfilment Identified by agreement * nro
Fact nro        Identified by Int
Fact signed     Identified by member * response

Act sign Actor member
  Related to response, date
  Holds when response // sign is available in states with responses
  Creates signed()
       , agreement() When response.respondent == member
                       && signed(response.offer.offerer, response)
       , agreement() When response.offer.offerer == member
                       && signed(response.respondent, response)

Act acquire Actor respondent
  Related to agreement, date
  Holds when agreement // acquire is available in states with signed agreements
  Creates fulfilment(agreement,
          Count(Foreach fulfilment' : fulfilment'.agreement == agreement))
```

Example – DMI Normative Control (1)

Only ‘ready’ members can engage with the transaction process:

```
Extend Act make-offer
  Conditioned by ready(offerer, date)

Extend Act respond
  Conditioned by ready(respondent, date)

Extend Act sign
  Conditioned by response.respondent == member || response.offer.offerer == member
               , ready(member, date)

Extend Act acquire
  Conditioned by agreement.response.respondent == respondent
               , ready(respondent, date)
```

And only respondents to signed agreements can acquire PDDs.

Adaptability example: Offerer can set custom policies

Certain offers can be for the purpose of research only:

```
Fact knowledge-institute Identified by member
Fact for-research        Identified by offer

Act require-research Actor offerer // setting a custom policy on an offer
  Related to offer
  Holds when offerer Conditioned by offer.offerer == offerer
  Creates for-research()

Extend Act respond // which is checked for all offers for research purposes
  Conditioned by knowledge-institute(respondent) || Not(for-research(offer))
```

Adaptability example: Offerer can set custom policies

Certain offers can be for the purpose of research only:

```
Fact knowledge-institute Identified by member
Fact for-research        Identified by offer

Act require-research Actor offerer // setting a custom policy on an offer
  Related to offer
  Holds when offerer Conditioned by offer.offerer == offerer
  Creates for-research()

Extend Act respond // which is checked for all offers for research purposes
  Conditioned by knowledge-institute(respondent) || Not(for-research(offer))
```

In which case the researcher gets the obligation to share findings:

```
Duty publish-research-findings Holder respondent Claimant offerer
  Related to agreement, date
  Violated when current-date > date + 10000 // within a year

Extend Act acquire
  Creates publish-research-findings(offerer = agreement.response.offer.offerer)
            When for-research(agreement.response.offer)
```

Example – Scenario

```
+membership-application(UvA, 20230101).
+membership-approved(UvA, 20230201).
+membership-fee-payment(UvA, 20230228).
+knowledge-institute(UvA).

+membership-application(Amsterdam, 20250101).
+membership-approved(Amsterdam, 20250301).
+membership-fee-payment(Amsterdam, 20250201).

+current-date(20260101).

+dossier(1).
make-offer(Amsterdam, Traffic, current-date, 1).
require-research(Amsterdam, offer(Amsterdam, Traffic, 1)).

respond(UvA, offer(Amsterdam, Traffic, 1), current-date).
 // ^-- violation if UvA is not a knowledge institute
sign(UvA, response(UvA, offer(Amsterdam, Traffic, 1)), current-date).
sign(Amsterdam, response(UvA, offer(Amsterdam, Traffic, 1)), current-date).

acquire(UvA, agreement(response(UvA, offer(Amsterdam, Traffic, 1))), current-date).
 // ^-- creates duty to publish research findings
```

eFLINT Design Decisions

The language has been developed to support:

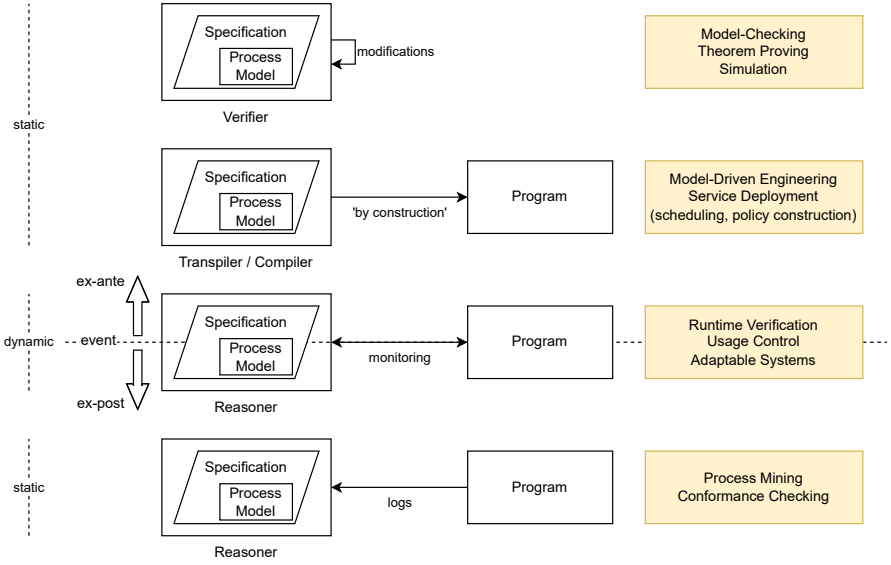
- Various types of reasoning: simulation, consistency-checking, completeness-checking, model-checking, monitoring, access control, (ex-post) compliance
- eFLINT specifications are inherently extensible, increasing adaptability
- eFLINT specifications are modular: new rules can be added/modified and connections between regulations and systems (interface) can be defined separately

Reflections on the design, applications and implementations of the normative specification language eFLINT

L. Thomas van Binsbergen[a], Christopher A. Esterhuyse[a], Tim Müller[a]

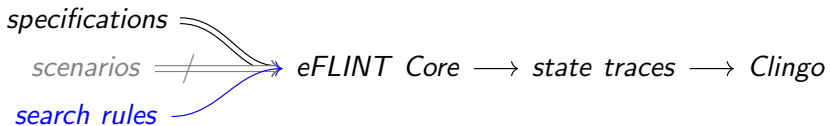
[a] *Informatics Institute, University of Amsterdam, Science Park 900, 1098 XH, Amsterdam, The Netherlands*

Approaches to Automating Compliance



Scenario Searching

Tweaked pipeline lets Clingo solve *scenario search* problems.



- The specification is input & translated as usual.
- There is no concrete scenario input.
  Instead, input rules define a *scenario search space* & criterion.

Clingo outputs a set of scenarios (one per answer).

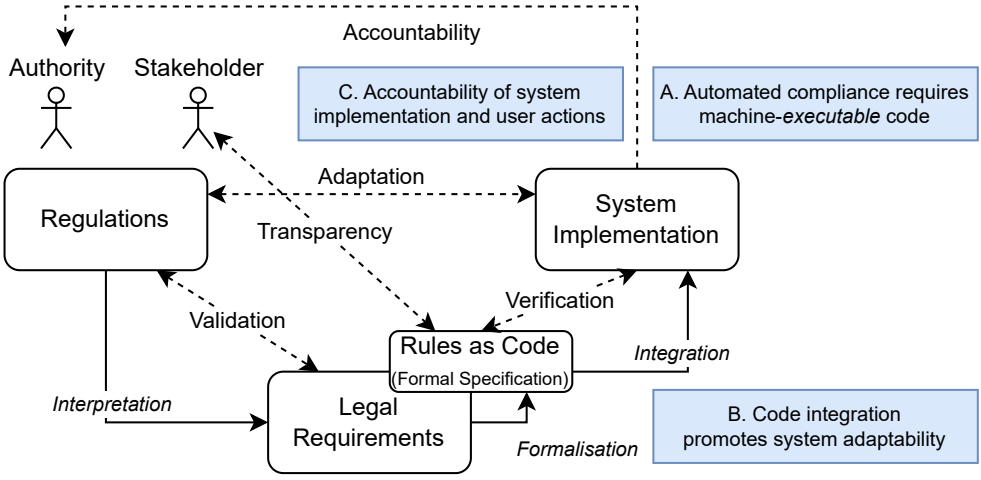
Model Checking

Examples of queries we can check:

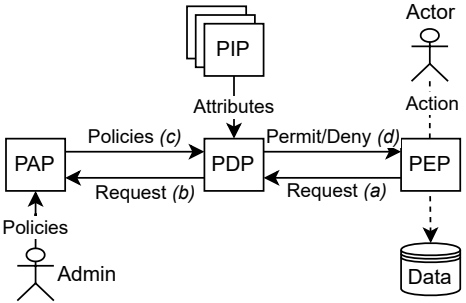
- Find me action-compliant scenarios that violate duties
- Find me duties that cannot be satisfied
- Find me duties that can only be satisfied through non-compliant actions
- Find me violated duties for which there is no compensation/resolution

B. Integration: Adaptability through Separation of Concerns

Overview

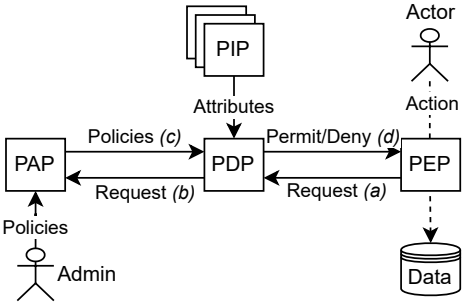

Accountability
Authority
Stakeholder
C. Accountability of system implementation and user actions
A. Automated compliance requires machine-*executable* code
Adaptation
Regulations
System Implementation
Transparency
Verification
Validation
Rules as Code
(Formal Specification)
Integration
Interpretation
Legal Requirements
Formalisation
B. Code integration promotes system adaptability

Access Control (AC) Basics – XACML architecture



- Request consists of *Actor*, *Action*, *Asset*
- Decision Point (PDP) executes policies to permit/deny actions based on attributes, such as offered PDD and current-date (requires a 'Context Handler'):

Access Control (AC) Basics – XACML architecture



- Request consists of *Actor*, *Action*, *Asset*
- Decision Point (PDP) executes policies to permit/deny actions based on attributes, such as offered PDD and current-date (requires a 'Context Handler'):

```
?Enabled(make-offer(<ACTOR>, <PDD>, <CURRENT-DATE>, <ASSET>))
?Enabled(response(<ACTOR>, offer(...,<ASSET>), <CURRENT-DATE>))
?Enabled(sign(<ACTOR>, response(...,<ASSET>), <CURRENT-DATE>))
?Enabled(acquire(<ACTOR>, agreement(...,<ASSET>), <CURRENT-DATE>))
```

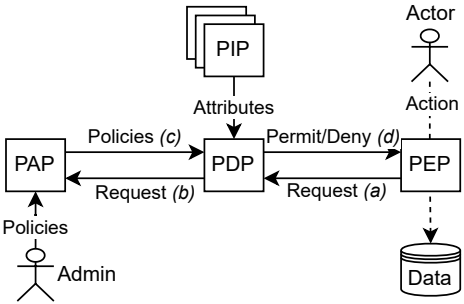
Example – Alternative: generating policy decisions

```
Fact asset          // identifies a dossier: offer, agreement, or fulfilment
Fact action         Identified by OFFER, REACT, SIGN, ACQUIRE // DMI interface
Fact decision       Identified by PERMIT, DENY

Fact policy-rule   Identified by actor * asset * action * decision
  Derived from
    policy-rule(member, dossier, OFFER, PERMIT)
        When Enabled(make-offer(member, pdd, current-date, dossier))
   ,policy-rule(member, dossier, RESPOND, PERMIT)
        When Enabled(respond(member, offer, current-date))
            && offer.dossier == dossier
   ,policy-rule(member, dossier, SIGN, PERMIT)
        When Enabled(sign(member, response, current-date))
            && response.offer.dossier == dossier
   ,policy-rule(member, dossier, ACQUIRE, PERMIT)
        When Enabled(acquire(member, agreement, current-date))
            && agreement.response.offer.dossier == dossier

?--policy-rule // generates the set of policy rules
```

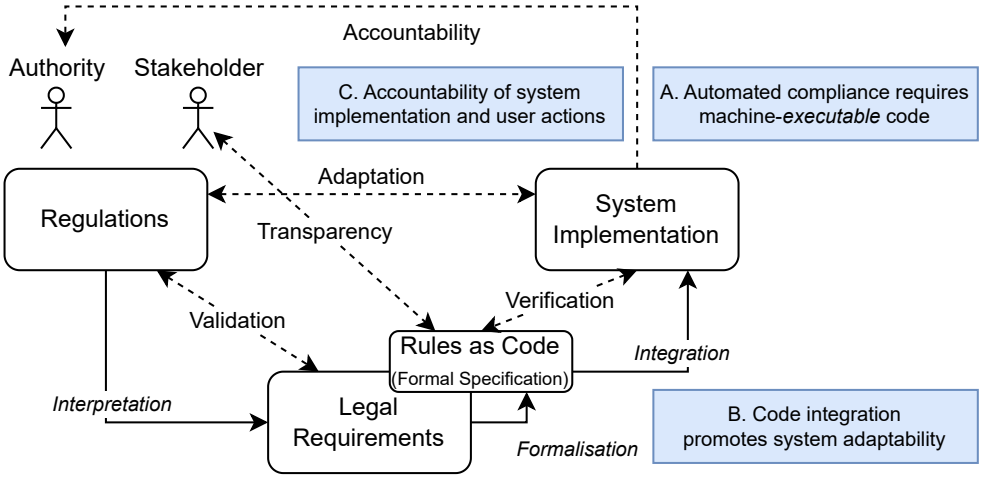
Reflections



- Access control as a form of Rules as Code with enforcement of policy rules.
- Requires agreement on semantics of actions and required attributes (Context Handler).
- Policies can be added modularly and can be more or less complex.
  For example, the requirement that the respondent is a knowledge institute.
  For example, authorisations based on GDPR reasoning (next).

C. Accountability of System and User Actions

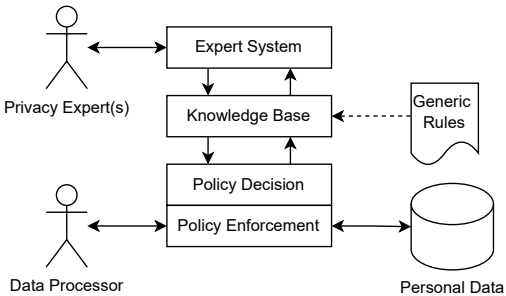
Overview


Accountability
Authority
Stakeholder
C. Accountability of system implementation and user actions
A. Automated compliance requires machine-*executable* code
Adaptation
Regulations
System Implementation
Transparency
Verification
Validation
Rules as Code
(Formal Specification)
Integration
Interpretation
Legal Requirements
Formalisation
B. Code integration promotes system adaptability

GDPR-based Access Control

Goal: Develop a knowledge-based, expert system for reasoning with GDPR-compliance and generating authorisations in distributed access and usage control implementations.

Approach: Develop case-generic rules once and for all and extend with case-specific input provided by users.



Is a given processing action lawful with respect to claimed legal bases in the GDPR?

GDPR-based Access Control

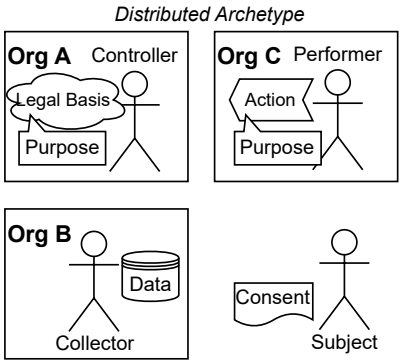
Is a given processing action lawful with respect to claimed legal bases in the GDPR?

GDPR-based Access Control

Is a given processing action lawful with respect to claimed legal bases in the GDPR?

Processing Action

- Identified by actor, asset, and *processing purpose*
- An action is always executed for one purpose



GDPR-based Access Control

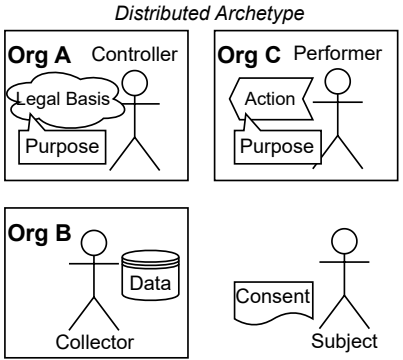
Is a given processing action lawful with respect to claimed legal bases in the GDPR?

Processing Action

- Identified by actor, asset, and *processing purpose*
- An action is always executed for one purpose

Legal basis

- Refers to Art. 6(1)(a-f), e.g., consent, legal obligation, legitimate interest, ...
- Identified by article (member) and *intended purpose*
- One or more legal bases can be *claimed*



GDPR-based Access Control

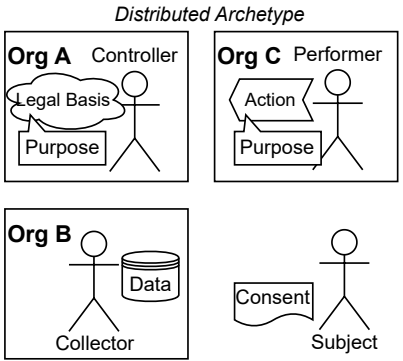
Is a given processing action lawful with respect to claimed legal bases in the GDPR?

Processing Action

- Identified by actor, asset, and *processing purpose*
- An action is always executed for one purpose

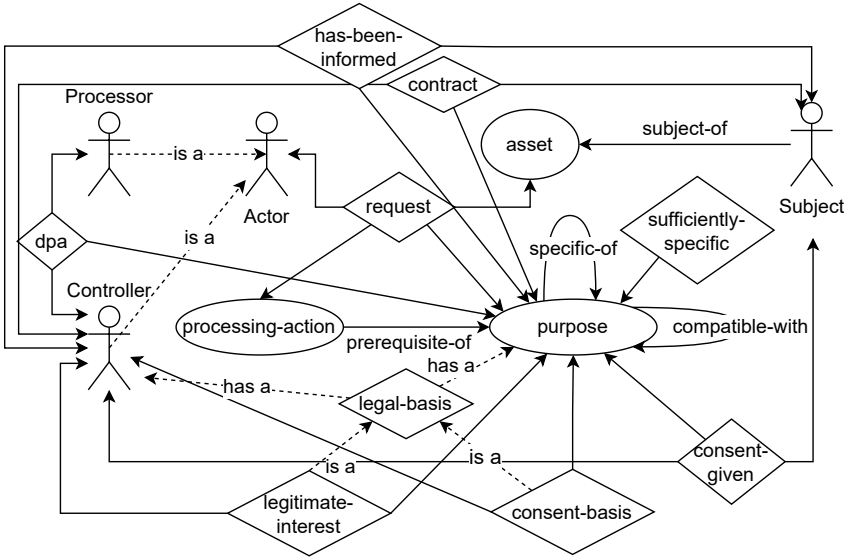
Legal basis

- Refers to Art. 6(1)(a-f), e.g., consent, legal obligation, legitimate interest, ...
- Identified by article (member) and *intended purpose*
- One or more legal bases can be *claimed*



Is a given collect action lawful with respect to claimed legal bases in the GDPR?

Ontology of lawful processing concepts



Knowledge base (ontology instantiations) and semantics

Case-generic rules for determining:

- Are the claimed legal bases valid?, e.g.
  - Is the intended purpose considered *sufficiently specific*?
  - Have the subjects been informed?
  - Have the subjects given consent (if legal basis is 'consent')?

$$\frac{\textit{legitimate-interest}(C, P) \quad \textit{sufficiently-specific}(P) \quad \forall_S(\textit{subject-of}(S, D) \rightarrow \textit{has-been-informed}(S, C, P))}{\textit{legal-basis}(C, P, D)}$$

Knowledge base (ontology instantiations) and semantics

Case-generic rules for determining:

- Are the claimed legal bases valid?, e.g.
  - Is the intended purpose considered *sufficiently specific*?
  - Have the subjects been informed?
  - Have the subjects given consent (if legal basis is 'consent')?
- Can the processing purpose and the lawful purpose be united? i.e.,
  - The processing purpose is *identical to* or *more specific than* the intended purpose
  - The processing purpose is *not incompatible* with the intended purpose

$$\frac{\textit{request}(U, A, P, D) \quad \textit{prerequisite-of}(A, P) \quad \textit{processor-for}(U, C, P') \quad \textit{specific-of}(P, P') \quad \textit{legal-basis}(C, P', D)}{\textit{lawful-request}(U, A, P, D)}$$

Knowledge base (ontology instantiations) and semantics

Case-generic rules for determining:

- Are the claimed legal bases valid?, e.g.
  - Is the intended purpose considered *sufficiently specific*?
  - Have the subjects been informed?
  - Have the subjects given consent (if legal basis is 'consent')?
- Can the processing purpose and the lawful purpose be united? i.e.,
  - The processing purpose is *identical to* or *more specific than* the intended purpose
  - The processing purpose is *not incompatible* with the intended purpose

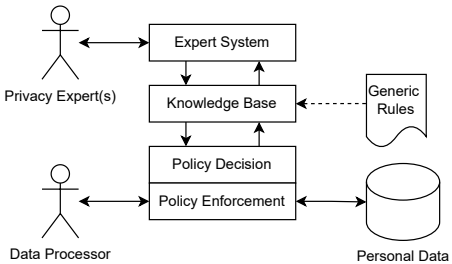
$$\frac{\textit{request}(U, A, P, D) \quad \textit{prerequisite-of}(A, P) \quad \textit{sufficiently-specific}(P) \quad \textit{processor-for}(U, C, P') \quad \textit{compatible-with}(P, P') \quad \textit{legal-basis}(C, P', D) \quad \forall_S(\textit{subject-of}(S, D) \rightarrow \textit{has-been-informed}(S, C, P))}{\textit{lawful-request}(U, A, P, D)}$$

Knowledge base (ontology instantiations) and semantics

Case specific statements

Expert drives the inference process by *claiming*:

- One or more legal bases
- Whether the intended purposes are sufficiently specific, more specific, not incompatible, etc.
- Whether (all) data subjects have been informed
- ... have given consent...
- etc.

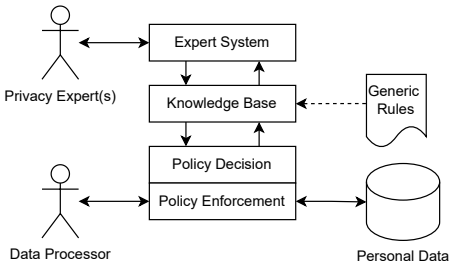


Knowledge base (ontology instantiations) and semantics

Case specific statements

Expert drives the inference process by *claiming*:

- One or more legal bases
- Whether the intended purposes are sufficiently specific, more specific, not incompatible, etc.
- Whether (all) data subjects have been informed
- ... have given consent...
- etc.



General approach

1. Encode case-generic rules in eFLINT and apply to all processing requests
2. Convert input by domain-expert into case-specific eFLINT statements
3. Assemble policy per request, make decision, and record inputs and outputs

Operationalisation within AMdEX-DMI

DMI·ECOSYSTEEM

```
Fact lawful-request
  Identified by actor * processing-action * purpose * asset
  Conditioned by request() // only considers created requests
                , prerequisite-of(processing-action, purpose)
```

$$\frac{\textit{request}(U, A, P, D) \quad \textit{prerequisite-of}(A, P) \quad \textit{specific-of}(P, P') \quad \textit{legal-basis}(C, P', D) \quad \textit{processor-for}(U, C, P')}{\textit{lawful-request}(U, A, P, D)} \tag{1}$$

```
Extend Fact lawful-request
  Holds when specific-of(purpose, purpose')
         && legal-basis(controller, purpose', asset)
         && processor-for(actor, controller, purpose')
```

Operationalisation within AMdEX-DMI

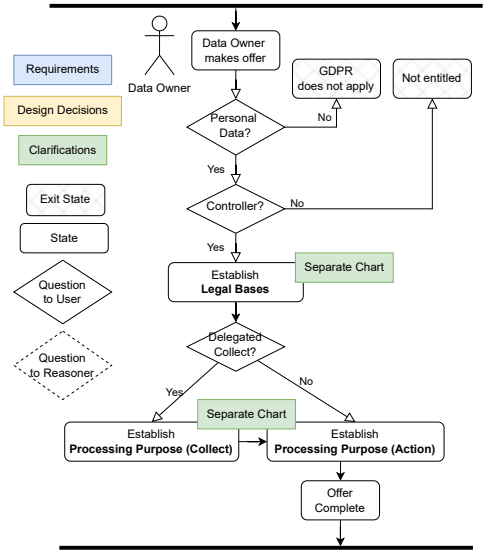
DMI·ECOSYSTEEM

```
Fact lawful-request
  Identified by actor * processing-action * purpose * asset
  Conditioned by request() // only considers created requests
                , prerequisite-of(processing-action, purpose)
```

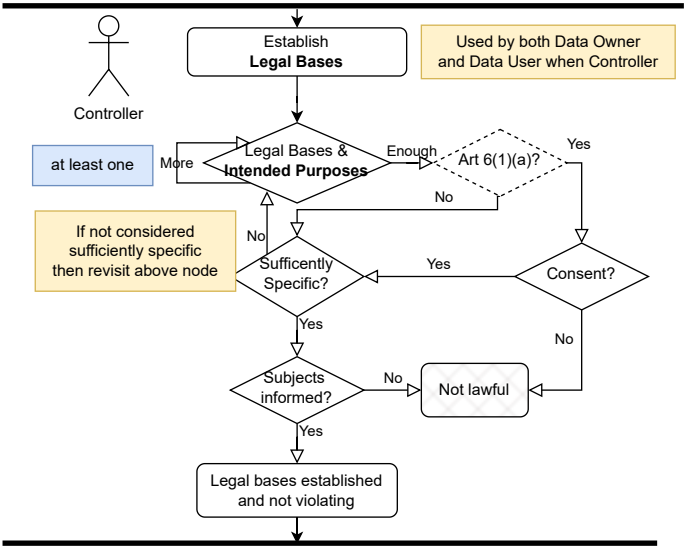
$$\frac{\begin{array}{c} \mathit{request}(U, A, P, D) \quad \mathit{prerequisite\text{-}of}(A, P) \quad \mathit{sufficiently\text{-}specific}(P) \\ \mathit{compatible\text{-}with}(P, P') \quad \mathit{legal\text{-}basis}(C, P', D) \quad \mathit{processor\text{-}for}(U, C, P') \\ \forall_S(\mathit{subject\text{-}of}(S, D) \rightarrow \mathit{has\text{-}been\text{-}informed}(S, C, P)) \end{array}}{\mathit{lawful\text{-}request}(U, A, P, D)} \tag{2}$$

```
Extend Fact lawful-request
  Holds when sufficiently-specific(purpose)
         && compatible-with(purpose, purpose')
         && legal-basis(controller, purpose', asset)
         && processor-for(actor, controller, purpose')
         && has-been-informed(subject, controller, purpose)
```

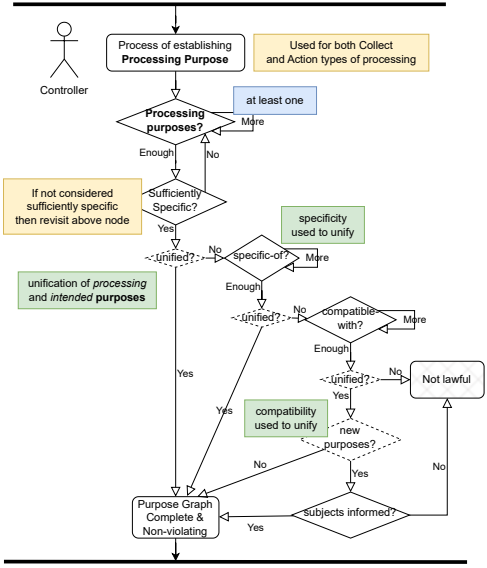
Operationalisation within AMdEX-DMI



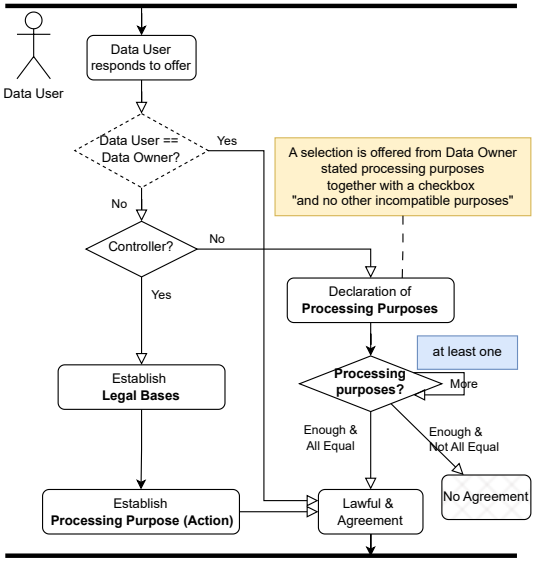
Operationalisation within AMdEX-DMI



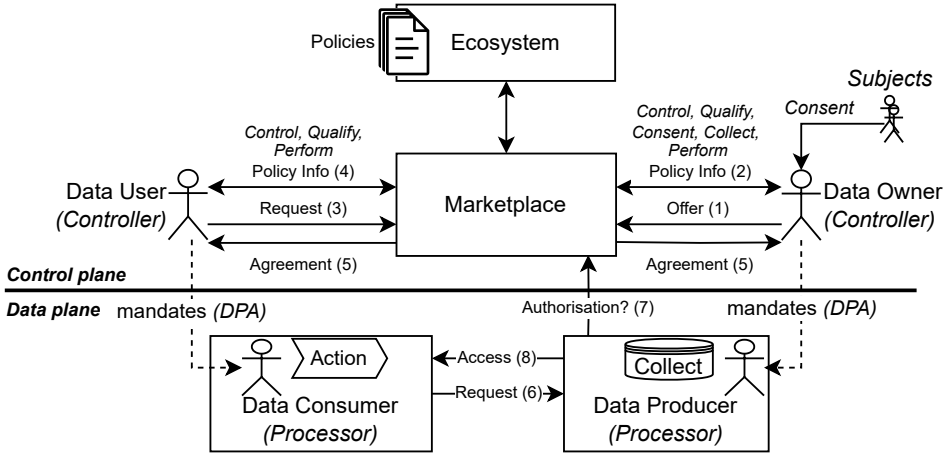
Operationalisation within AMdEX-DMI



Operationalisation within AMdEX-DMI

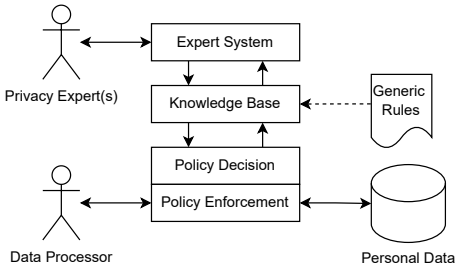


Operationalisation within AMdEX-DMI



GDPR-based Access Control

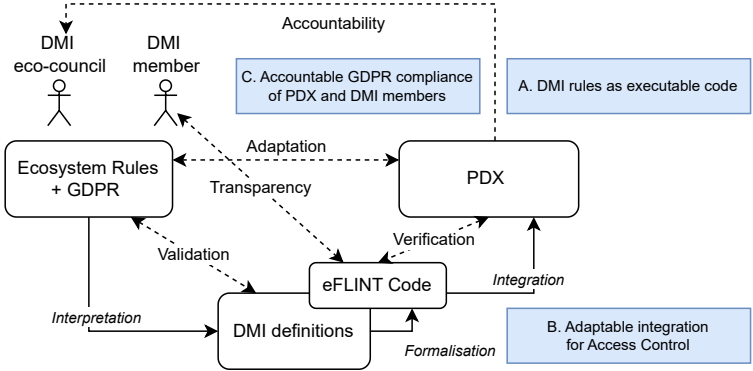
Goal: Develop a knowledge-based, expert system for reasoning with GDPR-compliance and generating authorisations in distributed access and usage control implementations.



Contributions:

- Raising the level of *abstraction* of policy specification to the level of the *domain-expert*.
  Before: System administrator sets (low-level) access policies
  After: Privacy expert submits claims regarding purposes and legal bases
- Authorisations are generated only when processing of legal data is lawful (according to the GDPR) in a *certifiable* and *accountable* manner
- Case-generic specification is *adaptable*, *extensible*, and *transparent*

Reflections



- The GDPR rules can be extended and modified, touching only the interface with the transaction process
- Certain parts of the DMI ecosystem rules can be modified as easily
- However, the transaction process itself is fundamental: modifying it requires changes across the board (possibly even to the DMI ecosystem rules themselves)

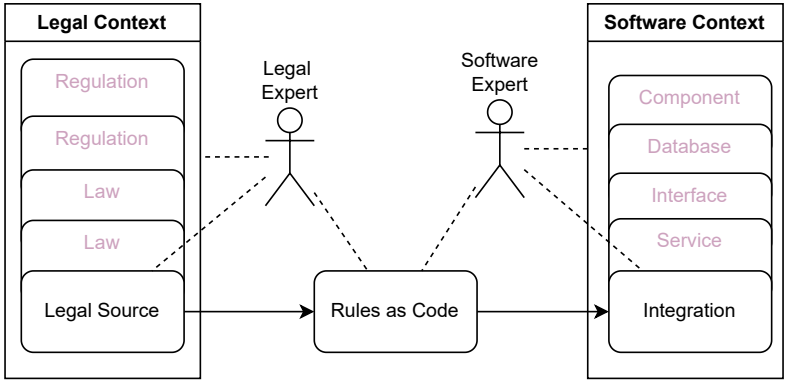
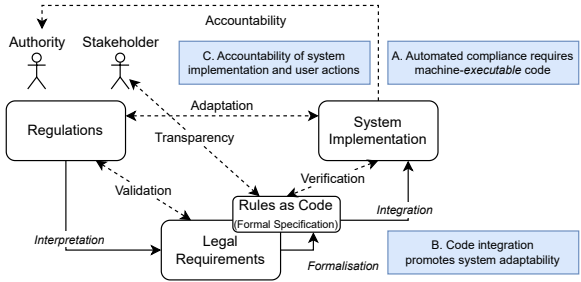


Figure: The interpretation process requires legal and software expertise



Takeaways

- Machine-*executable* code enables adaptable and accountable, compliant systems
- The formalisation process is instrumental in detection issues early on
- Integrate with separation of concerns for maximal adaptability and reuse
- Developing the rules as executable code requires legal *and* software expertise, e.g., defining the interface between system and code
- With the proposed semantics, various types of reasoning can be supported simultaneously

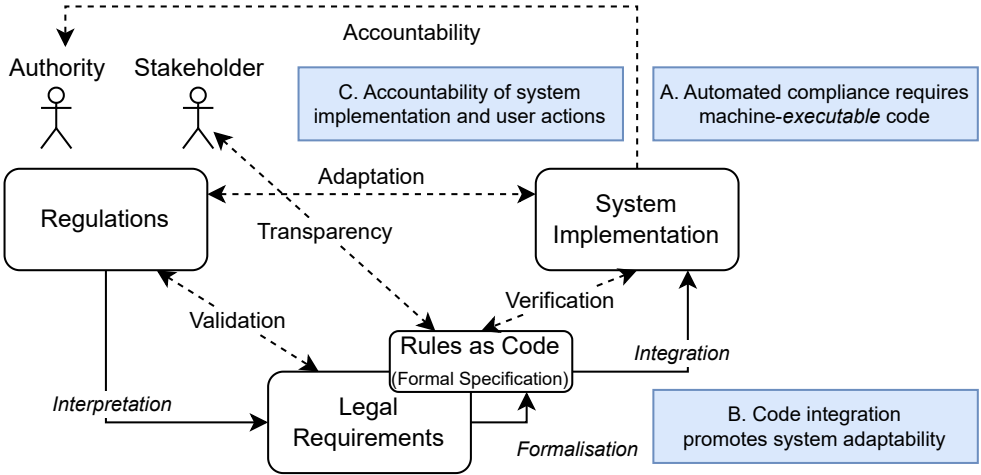
Rules as Executable Code for Adaptable and Accountable Software

L. Thomas van Binsbergen

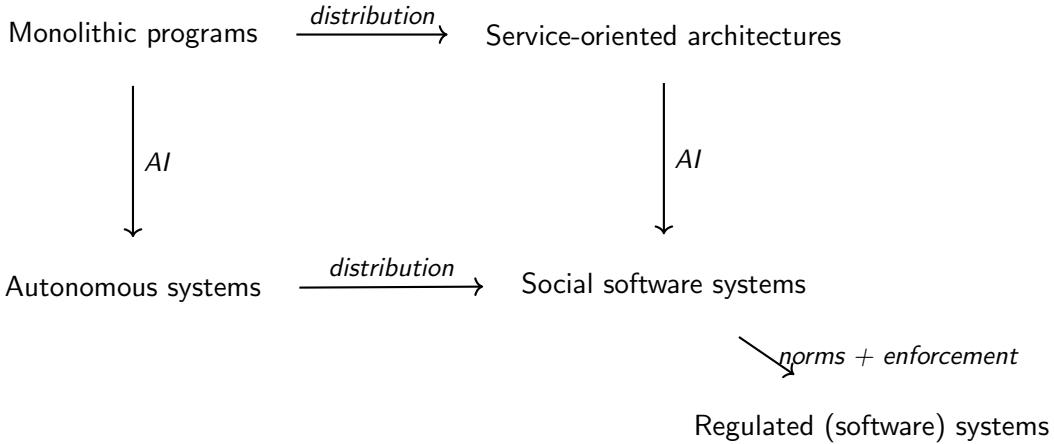
Informatics Institute, University of Amsterdam
l.t.vanbinsbergen@uva.nl

March 10, 2026

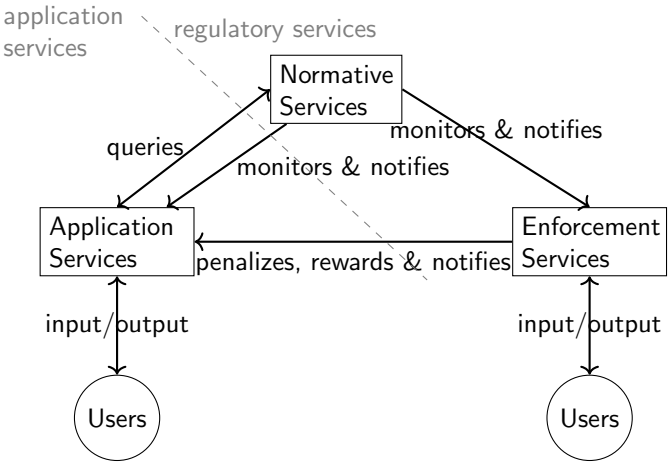
Recall: Verification of System Implementation



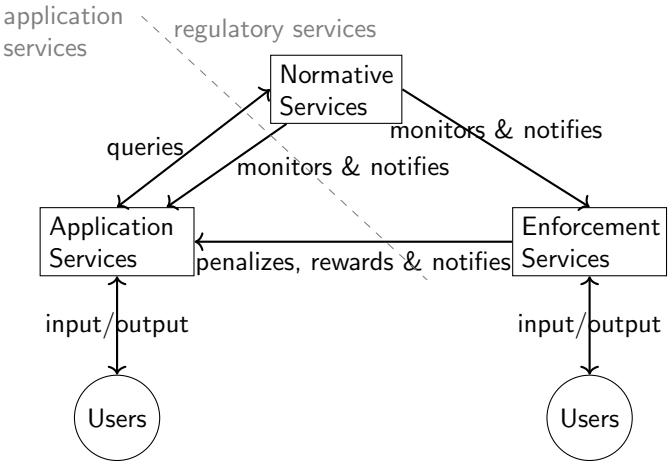
Towards regulated systems



Regulated system = application services + regulatory services



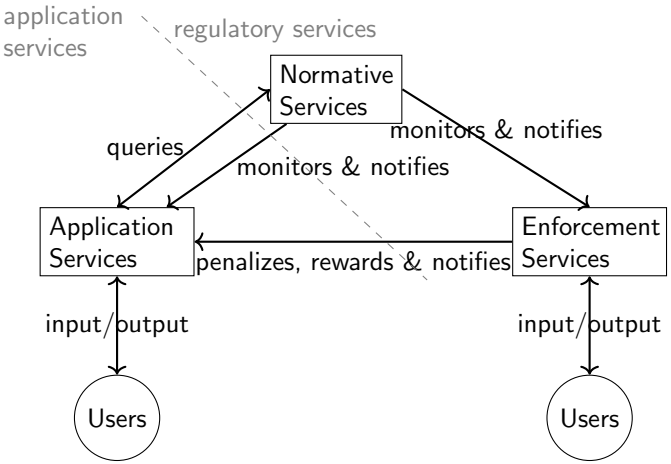
Regulated system = application services + regulatory services



Enforcement strategies

- Static, Ex-ante: orchestration and planning

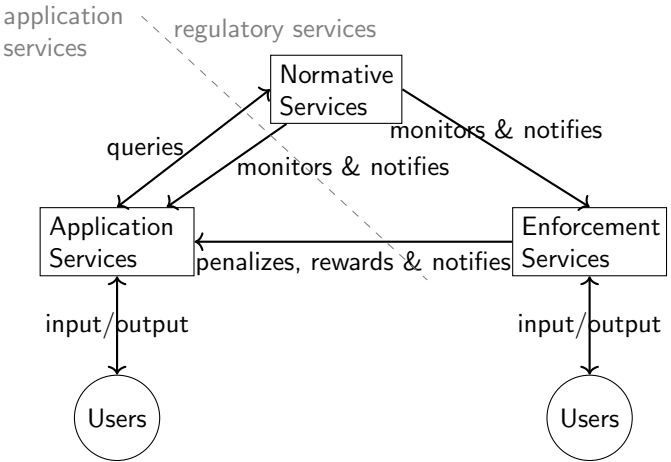
Regulated system = application services + regulatory services



Enforcement strategies

- Static, Ex-ante: orchestration and planning
- Dynamic, Ex-ante: access control

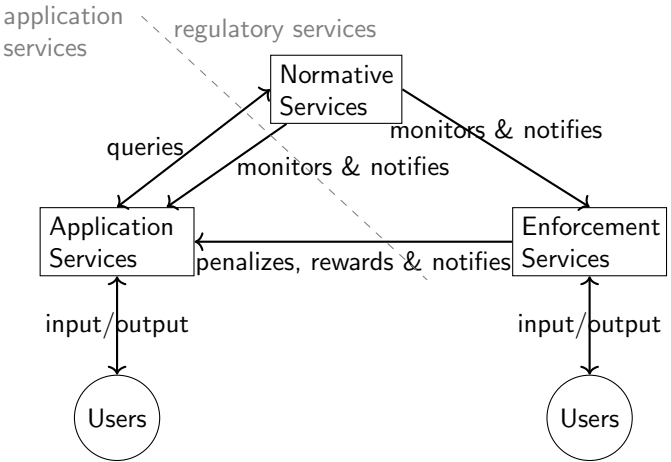
Regulated system = application services + regulatory services



Enforcement strategies

- Static, Ex-ante: orchestration and planning
- Dynamic, Ex-ante: access control
- Dynamic, Ex-post: usage control, runtime verification and adaptation

Regulated system = application services + regulatory services



Enforcement strategies

- Static, Ex-ante: orchestration and planning
- Dynamic, Ex-ante: access control
- Dynamic, Ex-post: usage control, runtime verification and adaptation
- Static, Ex-post: accountability and auditing

SSPDDP – Secure and scalable, policy-driven data exchange

CWI · VU University Amsterdam · Air France KLM · ABN·AMRO · ING

DL4LD – Data Logistics for Logistics Data

TKI Dinalog · NWO · Commit 2 Data · Thales · BiZZdesign · ciena · evofenedex · Gemeente Amsterdam · Universiteit van Amsterdam · TNO innovation for life · Airfrance KLM · Transfides · Oracle

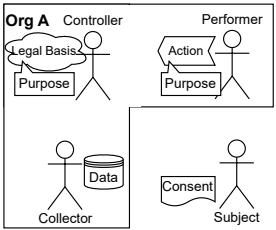
EPI – Enabling Personalized Interventions

NWO · St Antonius ziekenhuis · Parnassia groep · UMC Utrecht · Universiteit van Amsterdam · CWI · KPMG · EPI · Prinses Máxima Centrum · Philips · white box · SURF · Commit 2 Data · santeon · Ministerie van Economische Zaken · ZonMw

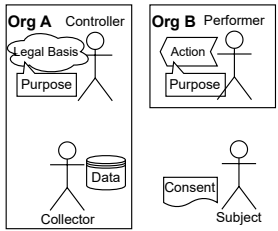
AMdEX – neutral data-exchange infrastructure

DMI·ECOSYSTEEM · Amsterdam Economic Board · SURF · amsix · Universiteit van Amsterdam · deXes · European Union, European Regional Development Fund, Investing in your future · Provincie Noord-Holland

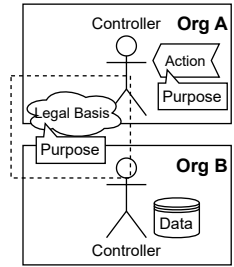
No Delegation Archetype



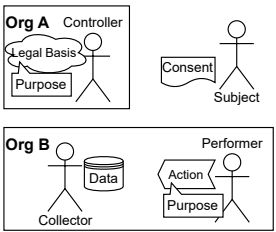
Delegated Action Archetype



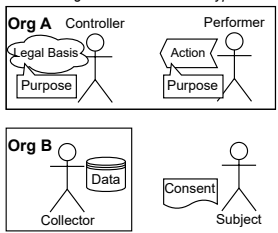
Joint Controllers Archetype



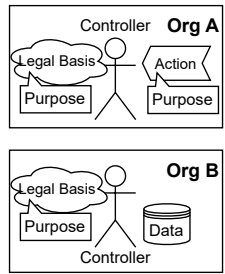
Delegated Processing Archetype



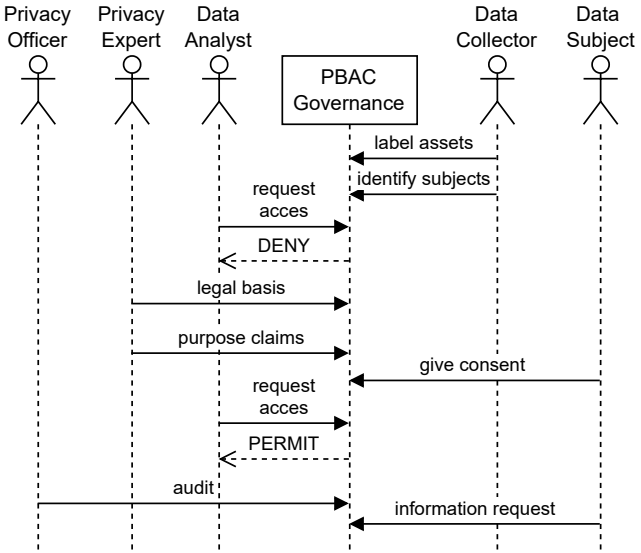
Delegated Collection Archetype



Independent Controllers Archetype



System interactions



Overview

1. Legal analysis
2. Ontology
3. Semantic specification (inference rules)
4. Semantic implementation (eFLINT)
5. Policy specification (purpose details, consent)
6. System integration (XACML, AMdEX)
7. Reflections

Legal Analysis (1)

Definition

A controller can claim a *legal basis* for processing for a specific intended purpose if the processing is lawful according to the GDPR (Art. 6), in which case one of the following applies:

- the data subject has given consent (Art. 6(1)(a)), or
- the processing is necessary for:
  - the performance of a contract with the data, or subject (Art. 6(1)(b)), or
  - compliance with a legal obligation (Art. 6(1)(c)), or
  - the vital interest of subject or natural person (Art. 6(1)(d)), or
  - public interest or vested authority (Art. 6(1)(e)), or
  - the controller has a legitimate interest (Art. 6(1)(f)).

And all data subjects involved must be informed about the legal basis and purpose, prior to the processing.

Legal Analysis (2)

Definition

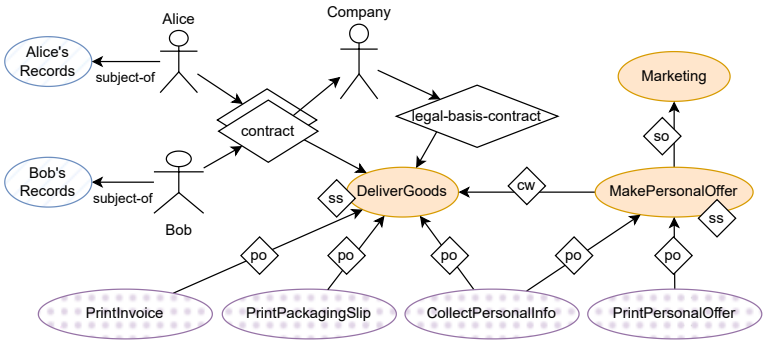
A *purpose-based processing request* connects an actor (a processor or controller) to a processing action, performed on an asset for a prescribed processing purpose. The request is considered lawful if:

- the action is prerequisite of the processing purpose, and
- the processing purpose is *sufficiently specific*, and
- the processing purpose:
  - coincides with a purpose that has a lawful legal basis, or
  - is more specific than a purpose that has a lawful legal basis, or
  - is not incompatible with a purpose that has a lawful legal basis.

Definition

A purpose is a *specific-of* of another purpose if it concretises a more abstract purpose without including elements not contained in the more abstract purpose.

Example purpose graph and scenarios



(a.) The processing actions that are prerequisites of delivering goods are lawful, for each individual subject, if a contract exists with that subject and for that purpose.

(b.) The further processing of the data to print and include a personal offer may be lawful depending on whether this purpose is considered to be incompatible with the delivery.

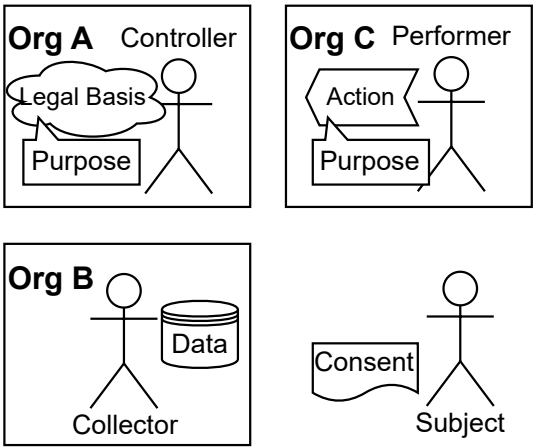
(c.) If, instead, the company asks for consent as a legal basis, the consent needs to state 'making a personal offer' and not 'marketing' as the latter is not deemed to be sufficiently specific.

Overview

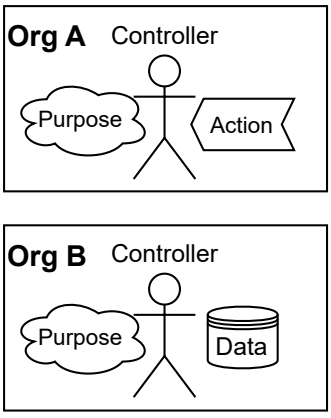
1. Legal analysis
2. Ontology
3. Semantic specification (inference rules)
4. Semantic implementation (eFLINT)
5. Policy specification (purpose details, consent)
6. System integration (XACML, AMdEX)
7. Reflections

Archetypical patterns of processing activities

Distributed Archetype



Independent Controllers Archetype



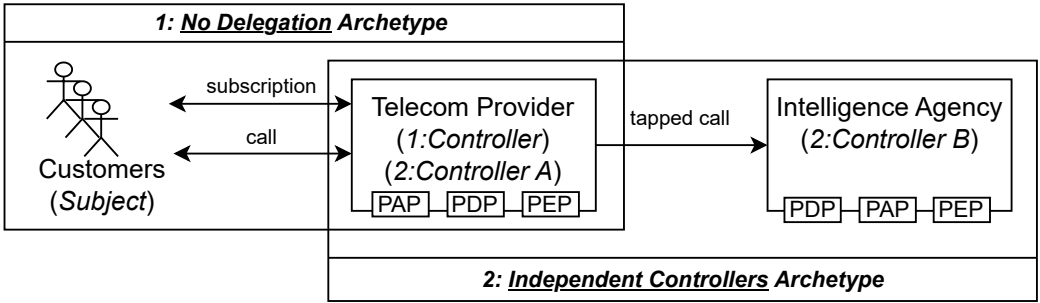
Policy administration capabilities and roles

Capability	**Policy (purpose-graph) contributions**	**Assigned to**
Control	legal-basis, dpa, has-been-informed, contract(s) (if applicable)	Controller, Authority
Qualify	prerequisite-of, compatible-with, specific-of, sufficiently-specific	Controller, Authority
Collect	asset(s), subject-of	Collector
Perform	request	Performer Collector
Consent	consent-given (including withdrawal of consent)	Subject

Policy administration capabilities and roles

Processing Archetype	Organisation	Policy Administration Roles
No Delegation	Controller	Controller, Collector, Performer
Delegated Action	Controller	Controller, Collector
	Performer	Performer
Delegated Processing	Controller	Controller
	Performer	Collector, Performer
Delegated Collection	Controller	Controller, Performer
	Collector	Collector
Distributed	Controller	Controller
	Collector	Collector
	Performer	Performer
Independent Controllers	Controller A	Controller, Collector
	Controller B	Controller, Performer

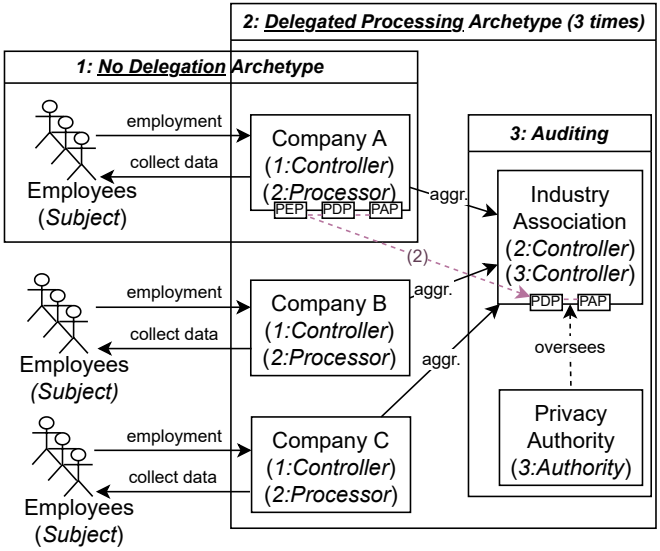
Example case: KPN and wiretapping



Scenario 2 checks:

- Upon sending: KPN's PEP confers with KPN PDP for *collecting*
- Upon receiving: Agency's PEP confers with Agency PDP for *performing*

Example case: industry benchmarking



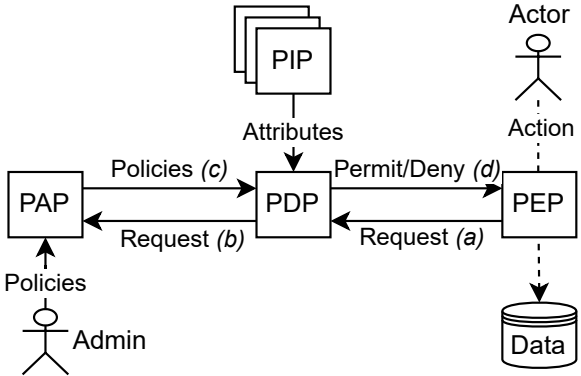
Scenario 1 checks:

- Company's PEP confers with local PDP for both collecting and performing (e.g., 'pay salary')

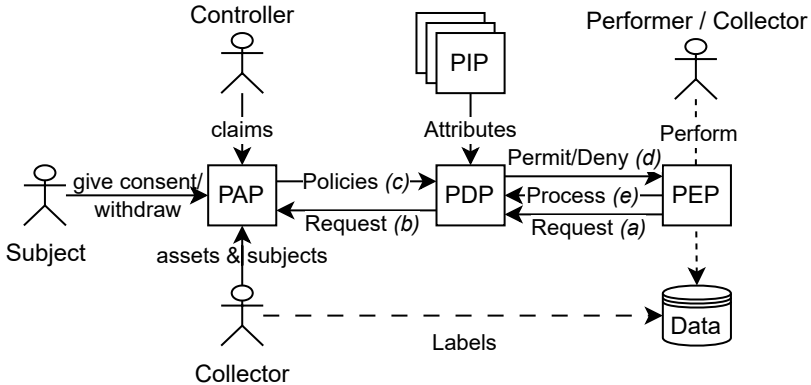
Scenario 2 checks:

- Company's PEP confers with Association's PDP for both collecting and performing (e.g., 'total salary, employee count')

Simplified XACML architecture (technical roles)

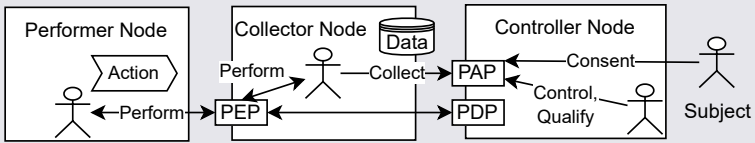


Simplified XACML architecture with PBAC policy administration



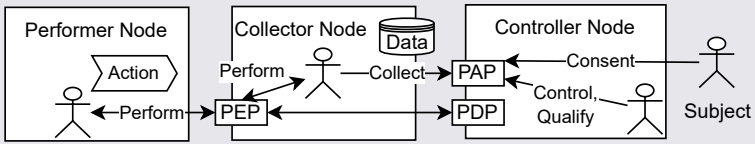
Mapping roles unto data exchange systems

Self-governed peer-to-peer system (distribution archetype)

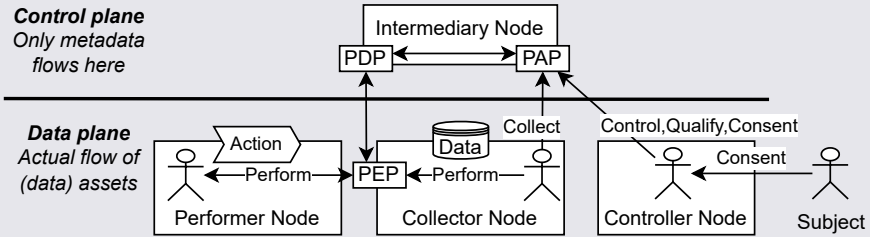


Mapping roles unto data exchange systems

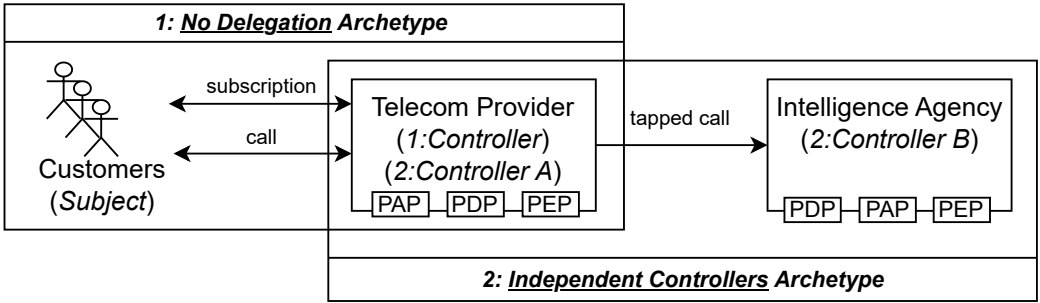
Self-governed peer-to-peer system (distribution archetype)



Peer-to-peer system governed by intermediary (AMdEX)



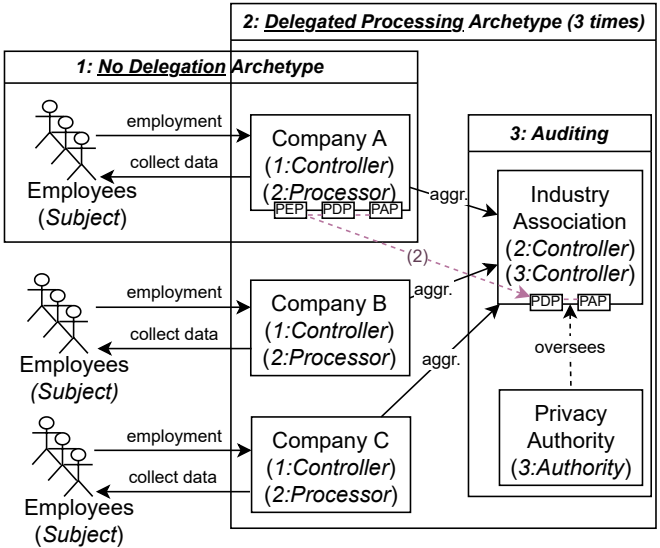
Example case: KPN and wiretapping



Scenario 2 checks:

- Upon sending: KPN's PEP confers with KPN PDP for *collecting*
- Upon receiving: Agency's PEP confers with Agency PDP for *performing*

Example case: industry benchmarking



Scenario 1 checks:

- Company's PEP confers with local PDP for both collecting and performing (e.g., 'pay salary')

Scenario 2 checks:

- Company's PEP confers with Association's PDP for both collecting and performing (e.g., 'total salary, employee count')

Reflections on accountability and explainability

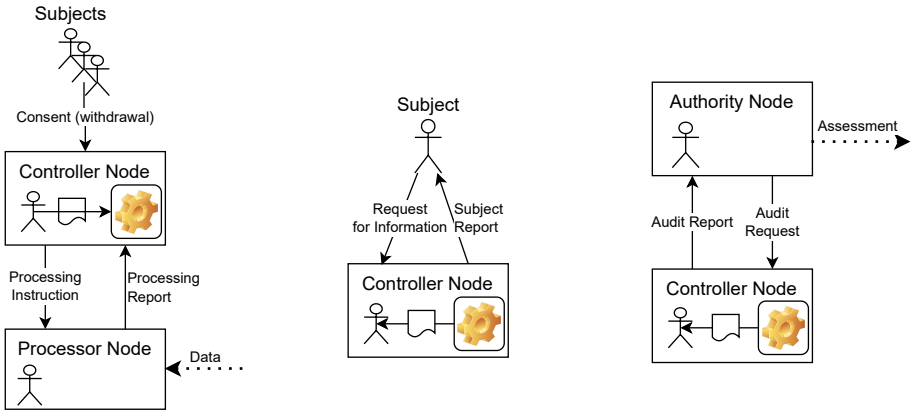


Figure: Different reasoning scenarios with different stakeholders.

Model Evolution

Reflected in current solution

- Original and further processing purposes need to be *sufficiently specific*
- Requirement to *inform subjects* of legal bases, prior to processing
  $\hookrightarrow$ which in some cases can be inferred
- Requirement to specify *processing purpose*

Necessary updates to be made

- Cases with two or more *independent controllers* (Control vs Perform capability)
- Cases with *joint controllership*

Future Work – AMdEX integration

We aim to show feasibility within the current AMdEX-DMI project.