Translating Usage Control Policies to Semantic Rules: A Model using OrBAC and SWRL

Nouha Laamech[a], Manuel Munier[a], Congduc Pham[b]

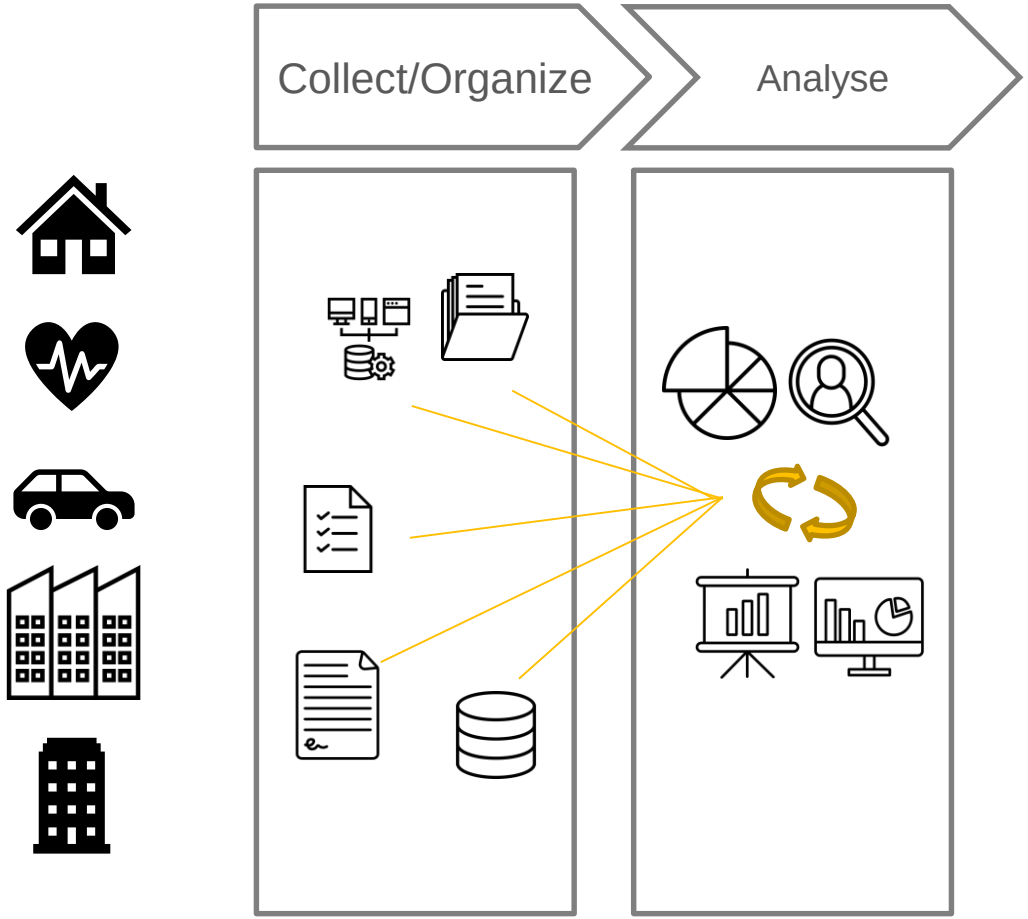
[a] Universite de Pau et des Pays de l'Adour, E2S UPPA, LIUPPA, Mont-de-Marsan, France
[b] Universite de Pau et des Pays de l'Adour, E2S UPPA, LIUPPA, Pau, France

27th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems
Athens, Greece, 06-08 September 2023

Summary

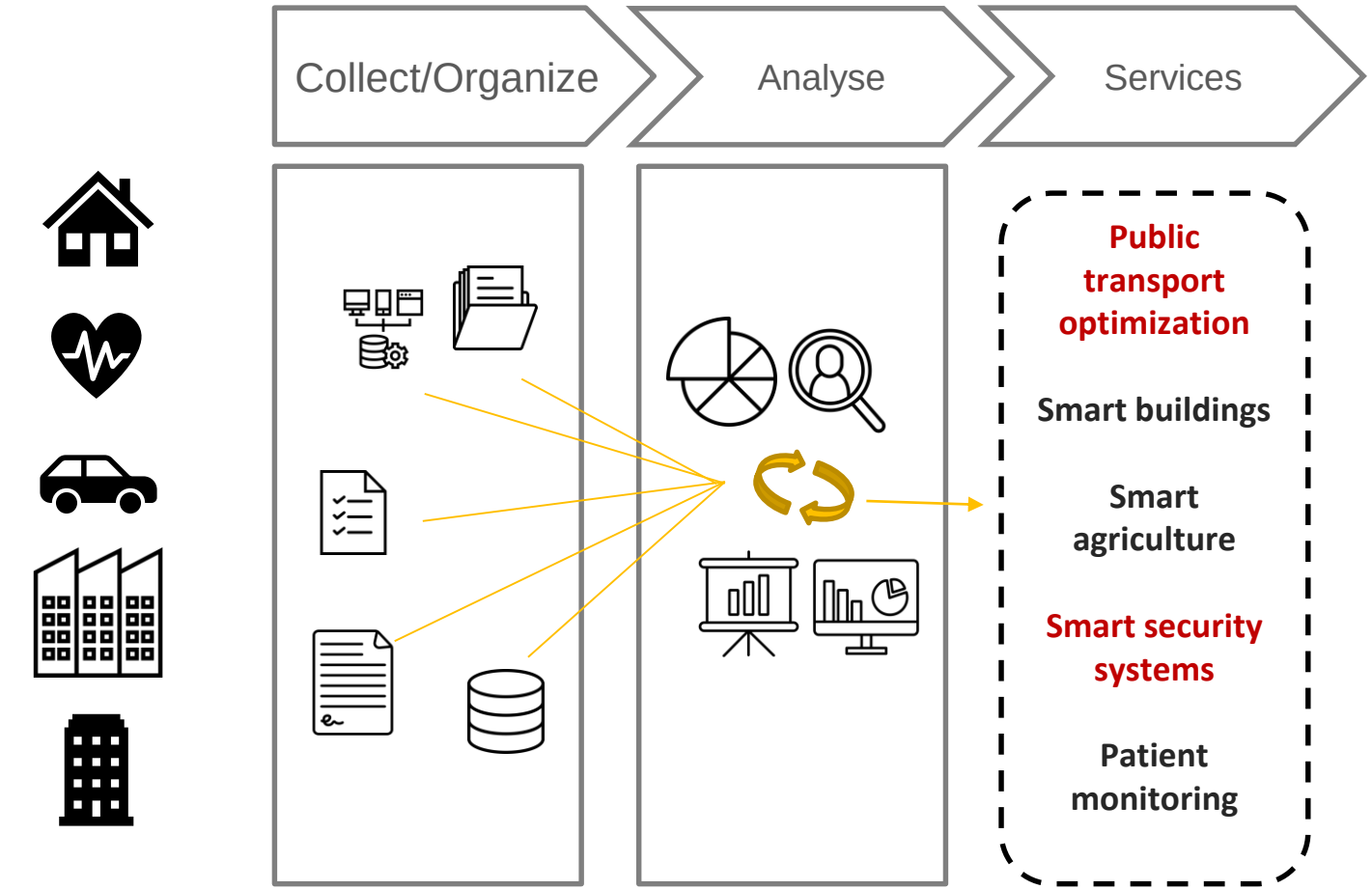
- Context
- Modelization process
- Proposed translation model
- Proof of concept and evaluation
- Conclusion and perspectives

Context



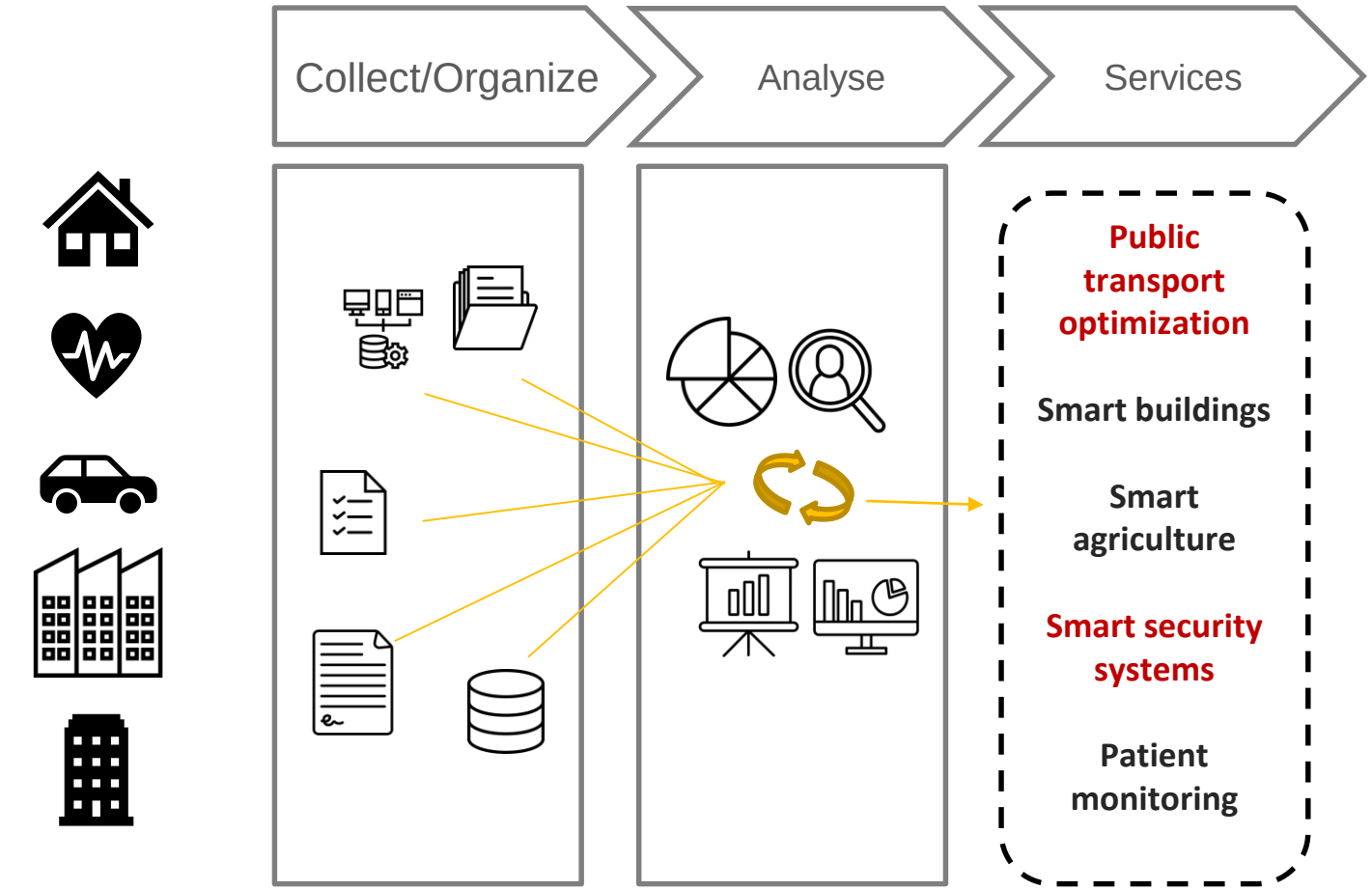
500 B objects
connected by 2030 [1]

Context



500 B objects
connected by 2030 [1]

Context



500 B objects connected by 2030 [1]

Challenges

(i) Security

(ii) Semantic interoperability

(iii) Context awarness

(iv) Consider obligation and prohibition

Security mechanisms

Access Control

- RBAC [2]
- ABAC [3]
- OrBAC [4]

Semantic web

- Ontolgies
- RDF
- SWRL[5]

Modelization Process

- Modeling a usage control model within OrBAC

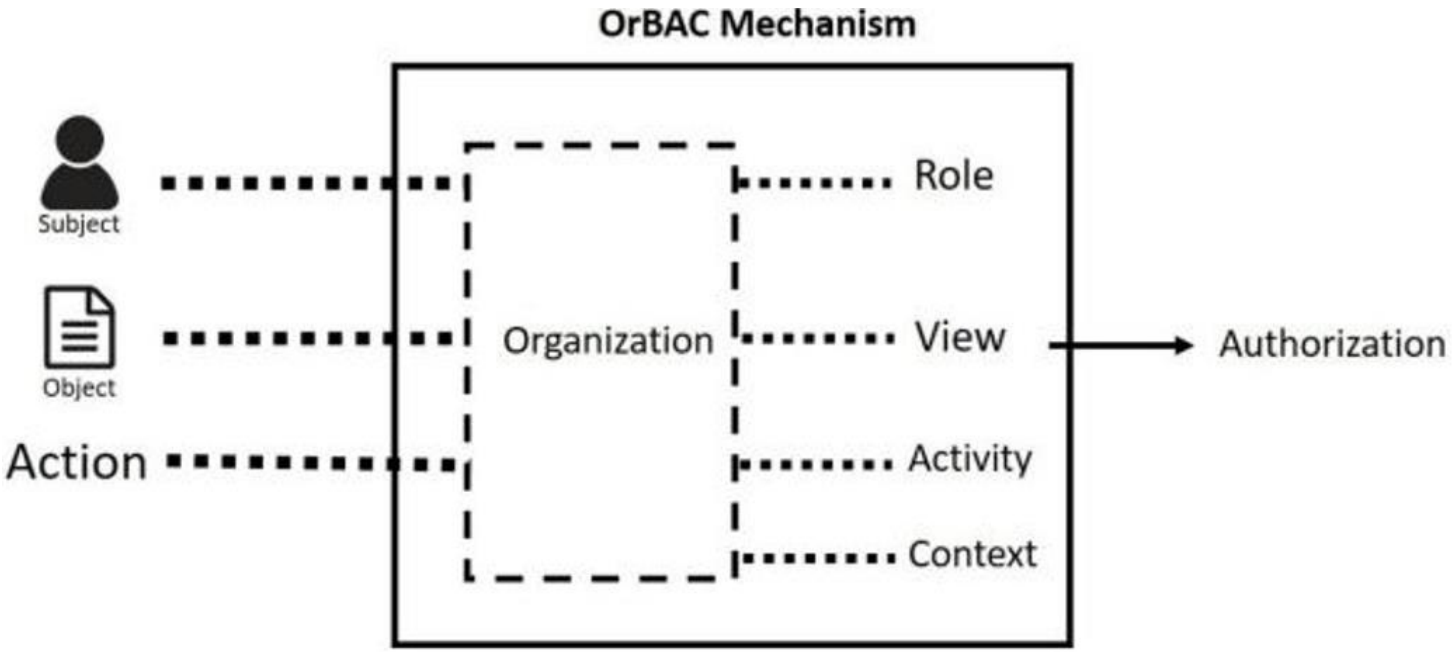


Fig. 1: OrBAC model

- Exemple of a usage control role

$$R1 \equiv Permission(\text{"}Data\_Producer\_Facility\text{"}, \text{"}farmer\text{"}, \text{"}access\text{"}, \text{"}soil\_humidity\text{"}, \text{"}delete\_after\_treatment\text{"})$$

Modelization Process

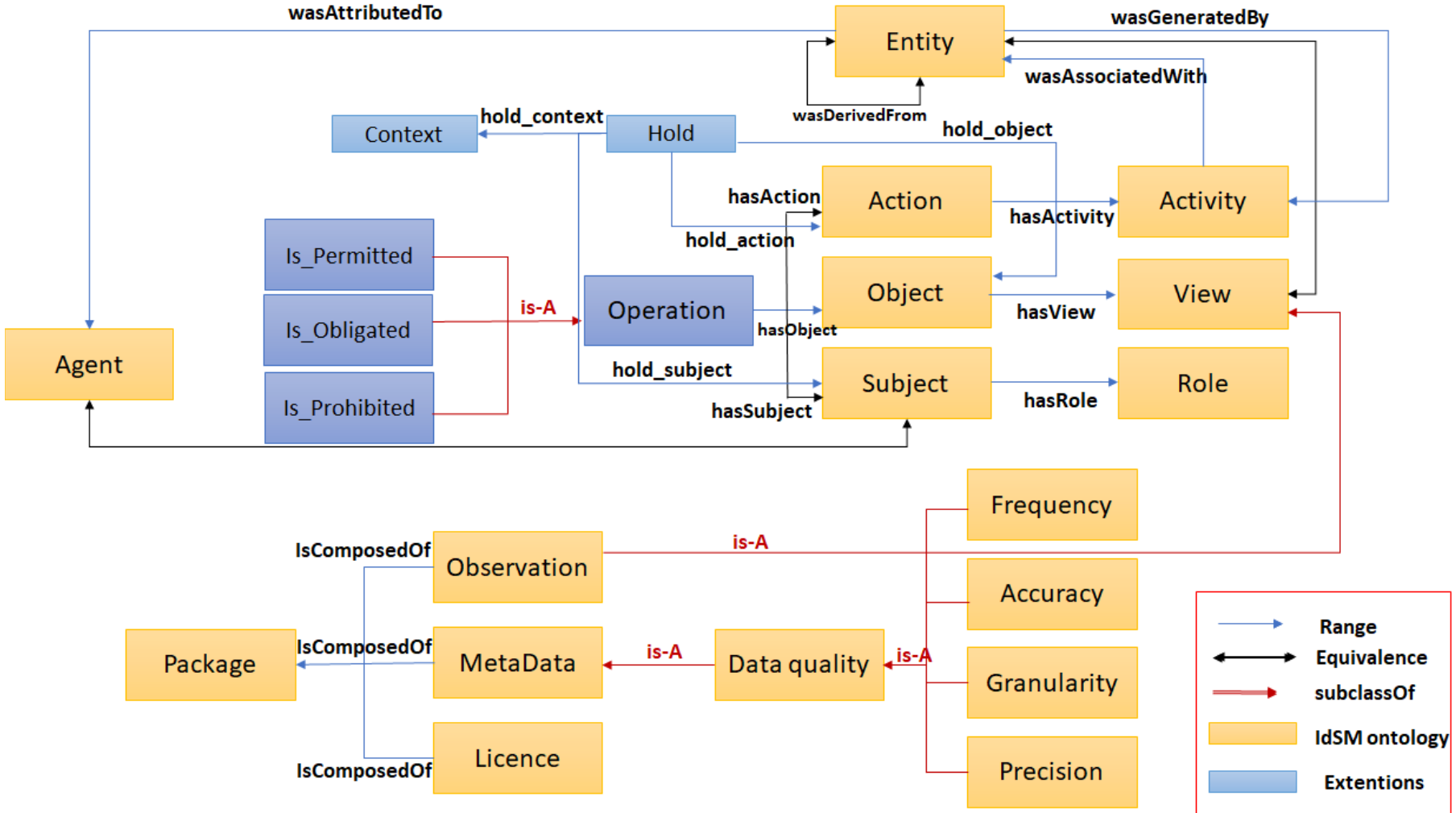


Fig. 2: IdSM Ontology overview

Proposed Translation Model

- Translation model: from OrBAC to SWRL

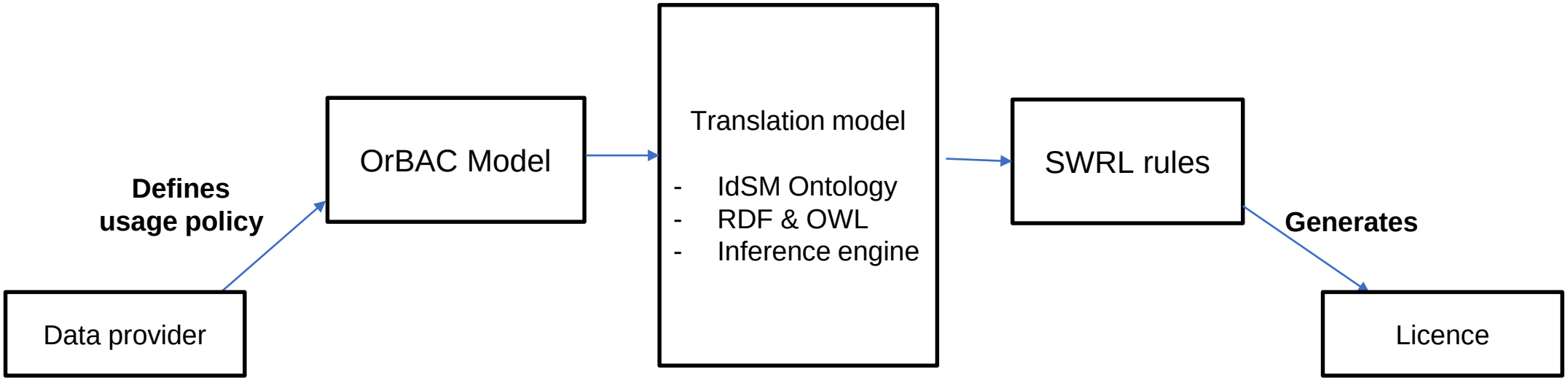


Fig. 1: Translation overview

Proposed Translation Model

- **From OrBAC to SWRL: context free**

R1 ≡ Permission(⟨**the_role**⟩,⟨**the_activity**⟩,⟨**the_view**⟩)

Rule = ”permission”
Role = ”**the_role**”
Activity = ”**the_activity**”
View = ”**the_view**”
Name_context = ” ”
Expression_context = ” ”

Rule = ”permission”
Role = ”**the_role**”
Activity = ”**the_activity**”
View = ”**the_view**”
Name_context = ” ”
Expression_context = ” ”

idsm:Operation(?op)

∧ idsm:hasSubject(?op, ?subject) ∧ orbac:empower(?subrole, ?subject)
∧ orbac:role_subrole(?role, ?subrole) ∧ orbac:Role(?role) ∧ sameAs(?role,⟨**the_role**⟩)

∧ idsm:hasAction(?op, ?action) ∧ orbac:consider(?subactivity, ?action)
∧ orbac:activity_subactivity(?activity,?subactivity) ∧ sameAs(?activity,⟨**the_activity**⟩)

∧ idsm:hasObject(?op, ?object) ∧ orbac:use(?subview, ?object)
∧ orbac:view_subview(?view,?subview) ∧ sameAs(?view,⟨**the_view**⟩)

→ idsm:is_Permitted(?op)

Proposed Translation Model

- **From OrBAC to SWRL: context-aware rules**

A- The definition of an OrBAC rule:

$$R1 \equiv \textbf{Permission}(\langle \textbf{no\_org} \rangle, \langle \textbf{the\_role} \rangle, \langle \textbf{the\_activity} \rangle, \langle \textbf{the\_view} \rangle, \langle \textbf{the\_context} \rangle)$$

B- Information retrieval and storage in JSON:

Rule = "**permission**"
Role = "**the_role**"
Activity = "**the_activity**"
View = "**the_view**"
Name_context = "**the_context**"
Expression_context = "**logical_conjunction** "

C- Context treatment:

idsm:Operation(?op)

∧ idsm:hasSubject(?op, ?subject) ∧ idsm:hasAction(?op, ?action) ∧ idsm:hasObject(?op, ?object)

∧ **logical_conjunction**

∧ swrlx:makeOWLThing(?h,?op)

→ orbac:Hold(?h) ∧ orbac:hold_subject(?h, ?subject)∧ orbac:hold_action(?h, ?action) ∧ orbac:hold_object(?h, ?object) ∧ orbac:hold_context(?h,**⟨the_context⟩**)

idsm:Operation(?op)

∧ idsm:hasSubject(?op, ?subject) ∧ orbac:empower(?subrole, ?subject)
∧ orbac:role_subrole(?role, ?subrole) ∧ orbac:Role(?role) ∧ sameAs(?role,**⟨the_role⟩**)

∧ idsm:hasAction(?op, ?action) ∧ orbac:consider(?subactivity, ?action)
∧ activity_subactivity(?activity,?subactivity) ∧ sameAs(?activity,**⟨the_activity⟩**)

∧ idsm:hasObject(?op, ?object) ∧ orbac:use(?subview, ?object)
∧ view_subview(?view,?subview) ∧ sameAs(?view,**⟨the_view⟩**)

∧ orbac:Hold(?h) ∧ orbac:hold_subject(?h, ?subject) ∧ orbac:hold_action(?h, ?action)
∧ orbac:hold_object(?h, ?object) ∧ orbac:hold_context(?h, **⟨the_context⟩**)

→ idsm:is_Permitted(?op)

Application: Smart Agriculture

A data provider is a farmer that wants his water meter observation to be accessible for all the farmers located in Europe.

Step 1 : OrBAC rule

R1 ≡ Permission(′Farm′,′ farmer′,′ access′,′ water_meter′,′ europe_context′)

Step 2 : JSON building

Role = ”farmer”
Activity = ”access”
View = ”water meter”
Name context = ”europe_context”
Expression context = ”hasLocation(?s,’Europe’)”

Application: Smart Agriculture

Step 3 : express context activation

idsm:Operation(?op) ∧ idsm:hasSubject(?op,?subject) ∧ idsm:hasAction(?op,?action) ∧ idsm:hasObject(?op,?object) ∧ idsm:hasLocation(?subject, ”Europe”) ∧ swrlx:makeOWLThing(?h,?op)
→ orbac:Hold(?h) ∧ orbac:hold subject(?h, ?subject) ∧ orbac:hold action(?h, ?action) ∧ orbac:hold object(?h, ?object) ∧ orbac:hold context(?h,”Europe context”)

Step 4 : final rule to grant permission

Idsm:Operation(?op) ∧ idsm:hasSubject(?op, ?subject) ∧ orbac:empower(?subrole, ?subject) ∧ orbac: role subrole(?role, ?subrole) ∧ orbac:Role(?role) ∧ sameAs(?role,”farmer”) ∧ idsm:hasAction(?op, ?action) ∧ orbac:consider(?subactivity, ?action) ∧ activity subactivity(?activity,?subactivity) ∧ sameAs(?activity,”access”) ∧ idsm:hasObject(?op, ?object) ∧ orbac:use(?subview, ?object) ∧ view subview(?view,?subview) ∧ sameAs(?view,”water meter”) ∧ orbac:Hold(?h) ∧ orbac:hold subject(?h, ?subject) ∧ orbac:hold action(?h, ?action) ∧ orbac:hold object(?h, ?object) ∧ orbac:hold context(?h, ”Europe context”)
→ is Permitted(?op))

Proof of concept and experimental evaluation

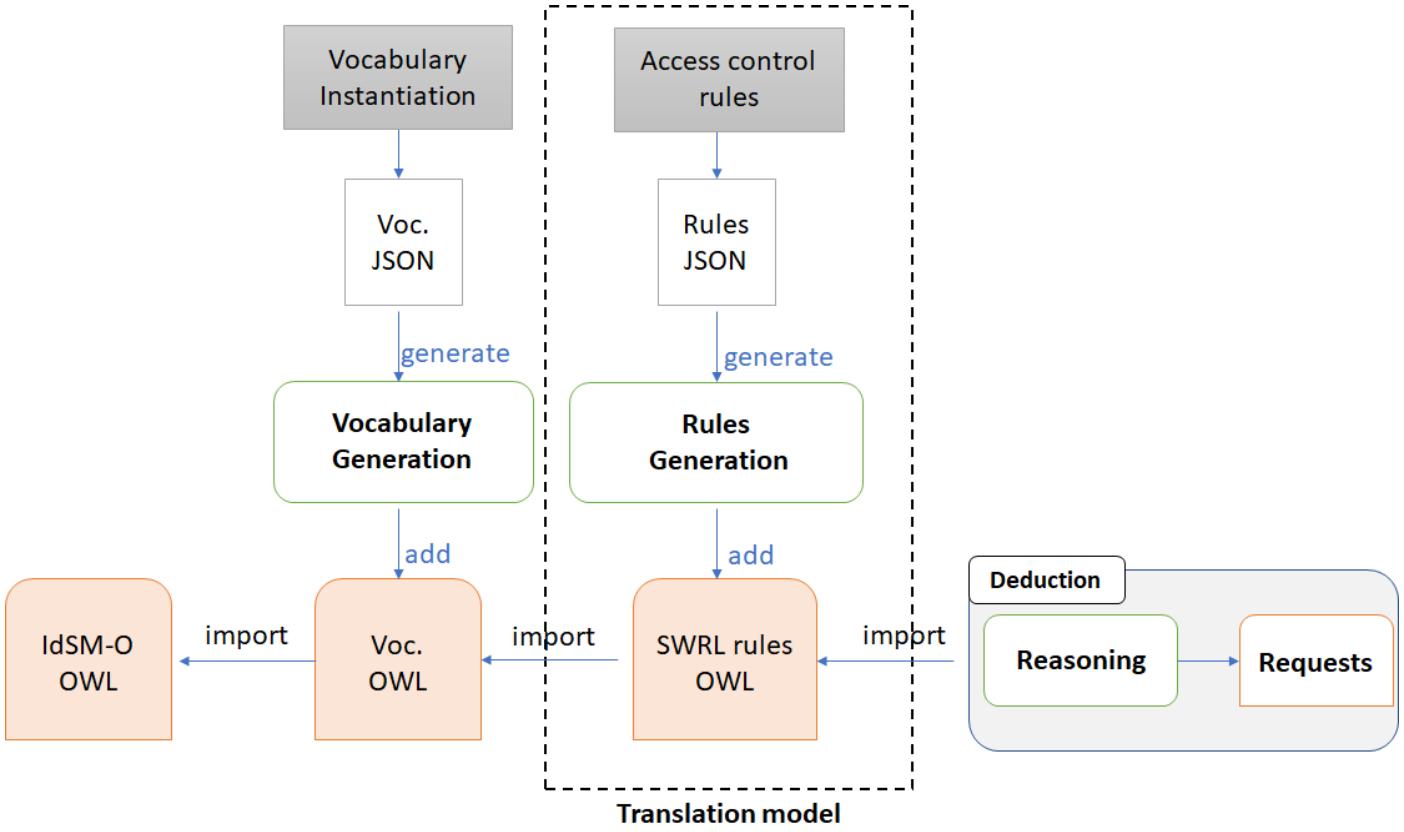


Fig. 3: Sequencing of the different modules

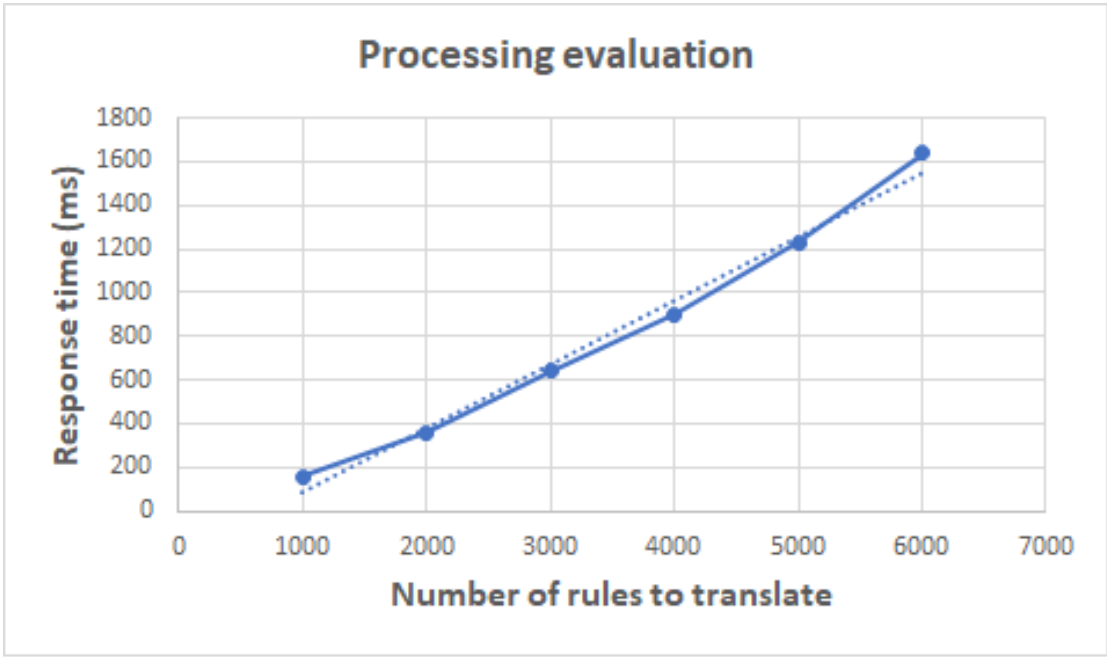


Fig. 4: Evaluation of the translating model

Conclusion

- Consideration of contexts during the translation process,
- Flexibility of the model that allows it to be applied in a wide range of domains,
- Incorporation of inheritance for more concise and modular policy definitions,
- Implementation of a three-layer proof of concept to manipulate the model translation.

Perspective

- In the context of rule conflict avoidance, it is necessary to establish rule priorities at an abstract level in order to effectively manage conflicts that may arise between different rules. This involves defining a hierarchy of rules based on their importance or relevance to the specific problem or domain being addressed,
- Consideration of additional parameters such as organizations and obligations rules during the translation model,
- Possibility to include other ontologies in addition to IdSM as the input for the first layer.

ACKNOLWEDGEMENTS

- This work is supported by the Conseil Départemental des Landes (PhD grant to N.Laamech).

REFERENCES

[1] Shantanu Pal, Ali Dorri, and Raja Jurdak. Blockchain for iot access control: Recent trends and future research directions. Journal of Network and Computer Applications, 203:103371, 04 2022.
[2] Neyadi, Darwish & Puthal, Deepak & Dutta, Joy & Damiani, Ernesto. (2023). Role-based Access Control in Private Blockchain for IoT Integrated Smart Contract.
[3] Yan, Liang & Ge, Lina & Wang, Zhe & Zhang, Guifen & Xu, Jingya & Hu, Zheng. (2023). Access control scheme based on blockchain and attribute-based searchable encryption in cloud environment. Journal of Cloud Computing. 12. 10.1186/s13677-023-00444-4.
[4] Cuppens Frederic and Miege Alexandre. Or-bac, organization based access control. The Review of Politics, Journees Druide, 2004.
[5] Ian Horrocks, Peter F Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, Mike Dean, et al. Swrl: A semantic web rule language combining owl and ruleml. W3C Member submission, 21(79):1–31, 2004.