A Systematic Approach toward Extracting Technically Enforceable Policies from Data Usage Control Requirements

Arghavan Hosseinzadeh, Andreas Eitel and Christian Jung
Fraunhofer IESE, Kaiserslautern, Germany

Keywords: Data Sovereignty, Data Usage Control, Industrial Data Space, MYDATA Control Technologies, Policy Classes, ODRL Policy Templates, Policy Specification, Policy Transformation, Policy Negotiation.

Abstract: Solutions for data sovereignty are in high demand as companies are willing to exchange their data in decentralized infrastructures. Data sovereignty is tightly coupled with data security and therefore, with data usage control policy specification. In this paper, we propose an approach to facilitate the processes of policy specification by data owners, policy transformation from a technology-independent to a technology-dependent language, and policy negotiation between the parties who exchange their data. We extracted an enterprise-relevant set of policy classes from the parties' security requirements in order to implement an editor that supports users in creating their machine-readable policies. Then, we developed an algorithm that benefits from the policy classes and constructs technology-dependent security policy instances. In addition, we proposed a policy negotiation approach which is based on the parameters of the extracted policy classes.

1 INTRODUCTION

The digital transformation affects many companies that are striving for industry 4.0. According to (Pauer et al., 2018), 79 percent of large enterprises and 70 percent of SMEs faced very strong changes due to digitization over the past five years. The authors state that data exchange between companies is an essential feature of digitization and data economy. Although data exchange is nothing new, it becomes more complex as companies are able to collect more data records in their silos and as they are going to link them. Data alone does not create any societal or economic added value, but if data is turned into information, they clearly bring an economic added value. This transformation can be achieved, for example, by enrichment, such as data analytics or by adding context information to data. Besides, data analytics enables companies to generate new information out of their existing data. Due to these possibilities, data have become an economic good in their own right. These changes enable companies to develop new business models and bring new products to the market. With all these changes, challenges and barriers arise during data exchange. Companies begin to fear that sensitive information (e.g., core data and business secrets) could be exposed (Pauer et al., 2018). In addition, companies find it problematic that they cannot control who will read their data (Pauer et al., 2018) and how it will be used. In order to address this issue, a joint committee of members from business, politics and research launched the Industrial Data Space (IDS) at the end of 2014 (Otto et al., 2016).

One of the core concepts of the IDS architecture is to provide a virtual data space for the secure decentralized exchange of data. IDS is based on a common governance model, thus provides data sovereignty for data owners. Realizing data sovereignty in such an architecture is a complex task and comes along with many challenges such as proper description of data access and usage restrictions and their technical enforcement on both data provider and data consumer sides. In the IDS context, a data provider (data owner) is an IDS party that offers his data on the IDS platform together with the terms and conditions of using that data; a data consumer is an IDS party that requests to use the provided data.

In general, policy managers can specify the data usage restrictions using various policy specification languages. These languages primarily differ in their degree of formalization. For instance, a policy may be expressed in natural language (e.g., "use my data only for billing and delete it after ten days") or in a more formal language (e.g., "purpose: billing, duty:

Hosseinzadeh, A., Eitel, A. and Jung, C.
A Systematic Approach toward Extracting Technically Enforceable Policies from Data Usage Control Requirements.
DOI: 10.5220/0008936003970405
In *Proceedings of the 6th International Conference on Information Systems Security and Privacy (ICISSP 2020)*, pages 397-405
ISBN: 978-989-758-399-5; ISSN: 2184-4356

delete after 10 days"). In fact, most organizations have high-level requirement specification expressed in natural language. Extracting the machine-readable policies from their requirements is a considerable challenge for them (Narouei et al., 2018). Moreover, the IDS Reference Architecture Model (Otto et al., 2018) does not impose a specific usage control technology and therefore, the IDS customers may use different technologies in order to enforce their security demands. Therefore, there is a need to describe the data usage restrictions in a technology-independent way. Besides, comparing policies is easier in a common machine-readable format than in natural language or in multiple technology-dependent languages. Therefore, the IDS stakeholders agreed on using the W3C standard Open Digital Rights Language (ODRL) (Iannella and Villata, 2018) to specify data usage restrictions in a technology-independent, machine-readable way, hereinafter referred to as "Specification Level Policy" (SLP). It is easily interpretable by the parties as well as their systems, and can later be translated to any technology-dependent language and eventually be enforced within the IDS parties' systems. An IDS party may use a usage control technology like MYDATA (Fraunhofer IESE, 2018a) that has its own expressive language (Fraunhofer IESE, 2018b). We call the policies that are specified in such a technology-dependent language, "Implementation-Level Policies" (ILP).

In a platform like IDS, a data provider specifies the provider-side and consumer-side SLPs. The provider-side policies are used to apply modifications on the data before it is transferred, and they control subsequent data access. The consumer-side policies impose usage restrictions on the data consumer systems.

This paper describes an approach to enhance the entire process of policy specification in a context such as IDS. The idea is to extract a set of policy classes from the stakeholders' requirements and to build a corresponding ODRL policy editor that assists stakeholders in specifying their SLPs. Policy classes are also used to derive the mapping rules that translate ODRL policies into MYDATA policies in a policy transformation service. As illustrated in Figure 1, a data provider can use the policy transformation service in order to semi-automatically translate his provider-side SLP (ODRL policy) to ILP (e.g., MYDATA policy) and then, enforce the generated MYDATA policy within his system. A data consumer can as well use the policy transformation service to transform the consumer-side SLP into an ILP, succeeding a policy negotiation process. This paper also explains how the policy classes facilitate the policy negotiation process between data providers and data consumers by clarifying the policy negotiation parameters.

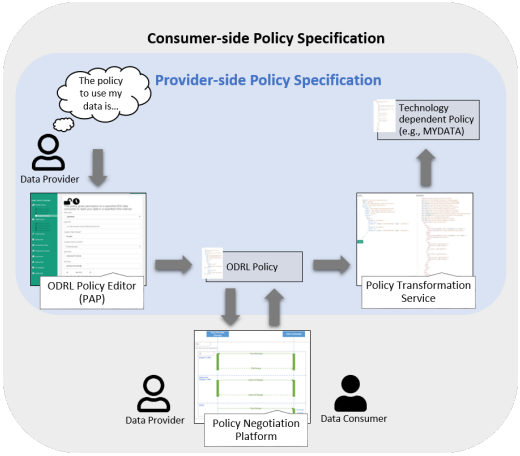

Figure 1: An ODRL policy editor is used to specify the provider-side and consumer-side ODRL policies. A policy transformation service translates them to a technology-dependent policy language such as MYDATA. A consumer-side policy has to be negotiated and confirmed by both data providers and data consumers before it is translated and enforced.

2 RELATED WORK

A policy editor can be used to facilitate the process of specifying ODRL policies. ODRL experts provide an ODRL policy editor that supports users in this task with a user-friendly interface (Lorenzo and Rodríguez-Doncel, 2018). The advantage of using such an editor is that the generated ODRL policies are validated and that users do not need to deal with the syntax of the language. However, this editor does not ensure that the generated policies reflect the user's actual demands. Therefore, they need to verify for themselves whether the generated policies express their usage control restrictions as intended.

Besides, Rudolph et al. suggested to collect Specification-Level Policy Templates (SLPTs), which are the blueprints for the SLPs, to generate a suitable security policy editor in natural language (Rudolph et al., 2016). Persuing their framework, the specification of ILPTs is a manual task that has to be performed by technology experts. Our method adopted this approach and extended it to enhance the specification of ILPs as well as the policy negotiation process.

According to (Pretschner and Walter, 2008), the element of *surprise* is a part of a negotiation among people and is unlikely to occur in an automated negotiation. It introduces new dimensions and valuations

over the lifetime of the negotiation process. In this paper, we refine the parameters that are negotiated with respect to the requirements in order to approximate our semi-automated negotiation process to a human-like approach.

3 POLICY SPECIFICATION

In the IDS context, policy specification is about defining security restrictions using the ODRL policy language and transforming them into ILPs such as a MYDATA policy. In this paper, we propose an approach to build an ODRL policy editor that not only validates the policies but also supports users by providing predefined templates that are specific to their requirements. As shown in Figure 2, the first step is the collection and refinement of all IDS use cases related to Data Usage Control. In a second step, the usage control requirements are derived and clustered as natural language constructs. Section 3.1 describes these constructs, also called policy classes, which form the basis to create ODRL policy templates. Next step is about mapping the policy classes to their corresponding ODRL policy templates. An exemplary ODRL policy template is explained in Section 3.2. Finally, the mapping rules from ODRL language to MYDATA language are extracted by studying these policy languages and with respect to the existing policy classes and consequently, a system that semi-automatically translates an ODRL policy template to a MYDATA policy template is developed. Overall, this approach can be generalized and also used for other ILPs.

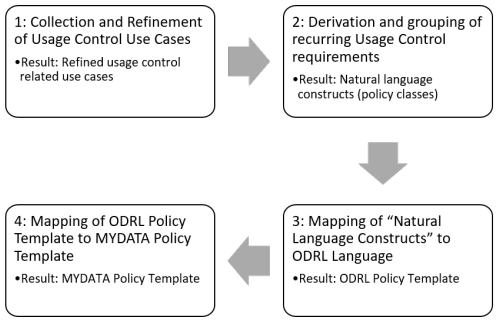

Figure 2: Policy specification approach.

3.1 Policy Classes

We collected Data Usage Control related use cases from various application scenarios such as Digital Supply Chain, Interconnected Enterprise Social Networks (ESN), Smart Urban Mobility, IDS Connectors and Intelligent Sensors. Many of the collected use cases reoccur (or can be applied) in different contexts. We manually assigned each use case to a more abstract class regardless of their context. Eventually, we ended up with a list of fourteen elaborated classes of policies:

1. Allow the usage of the data
2. Deny the usage of the data
3. Restrict the data usage for a group of users or systems
4. Restrict the data usage for specific purposes
5. Restrict the data usage when a specific event has occurred
6. Delete data immediately after (one) use
7. Modify data in transit
8. Modify data at rest
9. Allow data usage at most N times
10. Allow data usage only within a specific time interval
11. Log the data usage information
12. Notify a party when data is used
13. Share the data only under specific circumstances
14. Impose restrictions on the fine-grained actions

The use cases that only addressed the permission or prohibition of data usage were assigned to the first two policy classes. The use cases that restrict the usage of data to amount of usage or a specific time interval or depending on the receiver systems, usage purposes or occured events were assigned to the next distinguishable policy classes. Moreover, the use cases that focus on deleting data, modifying data, sharing data, logging information and notifying users were assigned to their corresponding policy classes. Finally, the use cases that impose specific restrictions to the fine-graned actions (e.g., restricting the resolution when printing data) were categorized as last defined policy class.

Although, these classes may be extended as IDS requirements grow, we were satisfied for the time being with supporting the policy classes mentioned above and creating respective ODRL policy templates.

3.2 ODRL Policy Language

The ODRL information model is the foundational basis for the ODRL policy semantic and describes the concepts, entities and their relationships (Iannella and Villata, 2018). ODRL policies represent not only permissions and prohibitions of actions over a data asset (as ODRL *rules*) but also additional restrictions

Listing 1: Sample ODRL policy template.

```
{
  "@type": ?,
  "uid":"http://example.com/policy:
      restrict-usage",
  "permission":[{
    "target": ?,
    "assigner": ?,
    "assignee": ?,
    "action":"use",
    "constraint":[{
      "leftOperand":"datetime",
      "operator":"gt",
      "rightOperand":{"@value": ?,
      "@type":"xsd:dateTime" }
     },{
      "leftOperand":"datetime",
      "operator":"lt",
      "rightOperand":{"@value": ?,
      "@type":"xsd:dateTime" }
    }]
  }]
}
```

and obligations as ODRL constraints and duties, respectively. The ODRL vocabulary describes the terms and expressions used in ODRL as well as how to encode them (Iannella et al., 2018). Both, the ODRL information model and the ODRL vocabulary serve as the core framework and represent the minimally supported expressiveness of ODRL language. ODRL profiles extend the vocabulary by new terms in order to enhance an ODRL policy with additional semantics.

As a second step of our policy specification approach (shown in Figure 2), we manually created the ODRL policy templates that represent the previously mentioned policy classes. Obviously, there are some values that need to be filled in by the policy manager. For example, Listing 1 represents an ODRL policy template for the policy class No. 10, "Allow data usage only within a specific time interval".

The type field of the template accepts the following values: *set*, *offer*, *request* and *agreement*. A *set* policy is defined by IDS infrastructure. An *offer* policy is offered by the assigner that denotes the data provider in IDS. A *request* policy is proposed by the assignee that denotes the data consumer in IDS, and an *agreement* policy is a policy that has been confirmed by both data provider and data consumer after a preceding negotiation process. In the IDS context, all ODRL rules of an *agreement* policy are defined by exactly one assigner and address exactly one specific assignee. After filling in the type, assigner and assignee values in the template, a policy manager only needs to add the target data asset for the policy as well as the start and end dates for his demanded time interval in order to instantiate the intended ODRL policy. The *datetime* left operand together with its corresponding type, *xsd:dateTime*, forms the time interval constraint placeholder. The given value for the start and end time will be added as right operands of these pre-defined constraints. In summary, the ODRL policy templates serve as a list of necessary ODRL actions, constraints and operators that are important in the IDS context.

3.3 ODRL Editor

It is not sufficient to only create a set of ODRL policy templates, since users still need further support in choosing the proper template and filling in the empty fields. Therefore, an ODRL editor is required. Such an ODRL editor is described in (Fraunhofer IESE, 2019). A policy manager only needs to choose a template from the classes that are listed on the left side of the editor, as shown in Figure 3, and fill in the required fields. The assigner field is automatically set to the unique ID of the data provider and the editor may provide more data, such as list of provided data and consumers, to assist the data provider in specifying the policy. Figure 3 also shows an instance of an ODRL policy created in the editor.

4 POLICY TRANSFORMATION

In order to transform an ODRL policy into a MYDATA policy, a set of mapping rules need to be applied. However, extracting a complete set of mapping rules is a rather complex task, yet we can use our policy classes to facilitate the derivation of the mapping rules required to transform SLP into ILP. In this section, we explain the MYDATA policy language and the algorithm for policy transformation.

4.1 MYDATA Policy Language

According to (Otto et al., 2016), data sovereignty enables the data owner to define the terms of use of his data assets. The Data Usage Control concept extends different approaches of policy enforcement, such as traditional Access Control mechanisms or Digital Rights and Trust Management, and in doing so, it realizes the data sovereignty concept of the IDS. Data Usage Control adds restrictions to data in order to control their future usage (Jung et al., 2014), (Kelbert and Pretschner, 2012), (Pretschner et al., 2008). In fact, it ensures that the receiver–after acquiring the data–utilizes the data as described in the

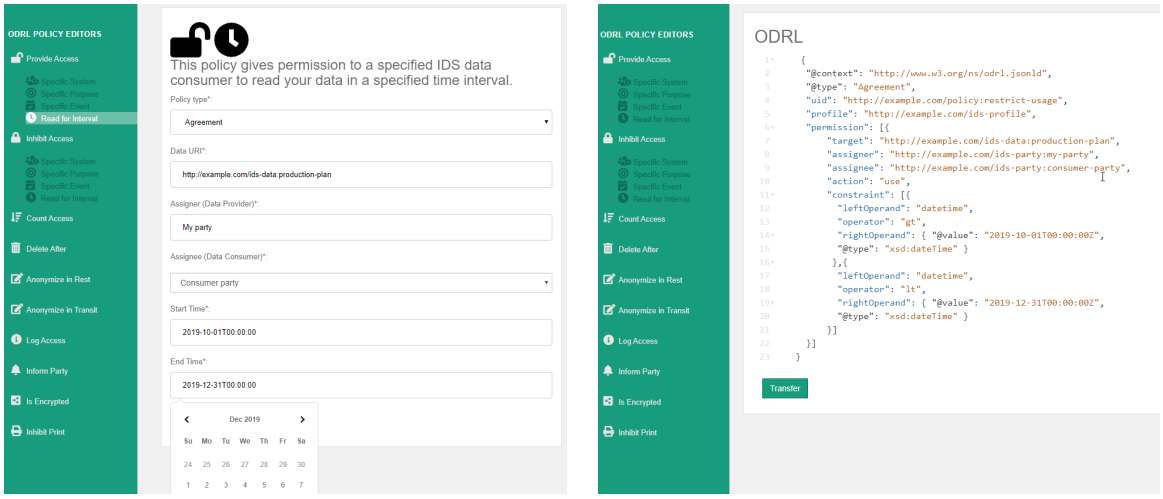

Figure 3: ODRL policy editor.

terms of use (Bussard et al., 2010). MYDATA Control Technologies is a technical implementation of Data Usage Control. It is based on the IND^{2}UCE framework (Fraunhofer IESE, 2018a) for Data Usage Control developed at Fraunhofer IESE. MYDATA uses Policy Enforcement Points (PEPs) to implement data sovereignty by monitoring or intercepting data flows. PEPs communicate with a Policy Decision Point (PDP) that utilizes policies and other components to make decisions. In addition, MYDATA provides a Policy Management Point (PMP) to manage all the components and policies. The ability to adapt the policies at any time makes the implementation of data sovereignty very flexible and adaptable to various (also future) usage scenarios in the IDS.

The MYDATA policy language (Fraunhofer IESE, 2018b) is an XML-based language to express restrictions on data usage. It represents a policy as a set of mechanisms that can describe the rules and actions of an ODRL policy based on monitored, intercepted or time triggered events. A MYDATA mechanism is an if-then-else statement in which the conditions are specified in the if-clause and the decision about allowing or denying the data usage is specified in the then- and else-clauses. The MYDATA policy language provides boolean and arithmetic logic as well as several functions (e.g., date and time) to express the conditions. In addition, it provides the connection to external systems, called Policy Information Points (PIPs), for information retrieval. It also provides execution of actions using Policy Execution Points (PXPs) and enforcement of data modification using MYDATA modifiers (Fraunhofer IESE, 2018b).

4.2 Mapping Rules

We use the ODRL policy templates, which are based on the policy classes, to create technology-dependent MYDATA policies by mapping ODRL policy actions, constraints and operators to MYDATA events, functions and operators. Algorithm 1 implements the required mapping rules receiving an ODRL policy as an input. In addition, it must be specified whether the policy is a provider-side policy or a consumer-side policy. This input will be given as a flag to later set the enforcement scope of the MYDATA policy, called MYDATA *solution*. By running the algorithm, a MYDATA policy is generated and a policy identifier is assigned to it. For each ODRL rule that is specified in the given ODRL policy, a corresponding MYDATA mechanism is created. Then, for all refinements and constraints of the given ODRL policy, a condition is defined and finally, the rule type (permission or prohibition) is specified as decision of the MYDATA policy.

We studied the policy classes to define the mapping rules for specifying the conditions. For each ODRL duty or constraint, an equivalent function in MYDATA policy language is used if available. Otherwise, the algorithm creates a general condition that is defined by a PIP. For example, in order to specify the ODRL constraint shown in Listing 2, the algorithm specifies a condition that uses a PIP to retrieve the purpose of usage and is satisfied when the purpose is ”marketing”.

Listing 2: ODRL Purpose Constraint.

```
"constraint":[{
  "leftOperand":"purpose",
  "operator":"eq",
  "rightOperand":{"@value":"Marketing
      ", "@type":"xsd:string"}
}]
```

Overall, in order to transform an ODRL policy into a policy that is specified in any other ILP than MYDATA, we can follow the same mapping rules and update the algorithm, correspondingly.

5 POLICY NEGOTIATION

A negotiation is a process that occurs between at least two parties about a certain subject and is driven by their demands (Bennicke et al., 2003). In a policy negotiation process in the IDS, the subject is the Data Usage Control policy, and the demands reflect the involved parties' security and privacy requirements. Thus, an agreed policy is the outcome of the process.

Data consumers may have different data usage preferences and system architectures than data providers and use different policy enforcement technologies. Therefore, they may request changes on the consumer-side policies that are published by the data providers on the IDS platform. In the context of the IDS, an *offer* policy addresses the usage policy of a specific data asset. A data consumer evaluates whether it is possible to technically enforce the *offer* policy at the data consumer side and whether the policy complies with his demands. For example, when the data consumer requires to use the data in a specific time interval, the policy needs to be evaluated to check whether it is offering the usage of data within that desired time slot. After the evaluation, either the data consumer entirely agrees to the conditions specified in the offered policy, or it requires to slightly change the policy. Consequently, the data consumer creates a suitable *request* policy, attaches it to a request message and sends it to the data provider. The policy negotiation process is initiated as soon as the data consumer sends the request message to the data provider as shown in the Figure 4. The data provider evaluates the received *request* policy and creates a corresponding new *offer* policy and send it back to the data consumer. The parties may build their own *policy evaluation components* that automatically evaluate the received policies and generate the corresponding new *offer* or *request* policies. The policy negotiation process continues until the data provider and data consumer agree on a Data Usage Control policy that addresses their needs and can be enforced within their systems. When the *request* policy matches the *offer* policy, the data provider creates an *agreement* policy, and sends it to the data consumer as an agreement message. Next, the data consumer has to acknowledge the *agreement* policy. The data provider can always send a reject message as a reply to a request message and thus require to terminate the policy negotiation process.

It might be a drawback of an automated policy negotiation process that the data consumer can create new *request* policies aiming to achieve an agreement with the least restrictions. However, the policy negotiation process is not necessarily about achieving an agreement that minimizes the security restrictions offered by the data provider, but it is about achieving an agreement that fits best to both parties' demands and the technical feasibility of the enforcement technologies. In fact, the demands may have inverse correlations. For example, a data provider may accept that the data consumer uses the data in a different system and for different purposes than originally offered, but may in return insist on a shorter usage period. However, the details of the policy negotiation process, such as where the *policy evaluation component* is located or the mode of operation of the time-out handling mechanism, is not the focus of this paper, but rather the policy negotiation approaches and their parameters.

5.1 Policy Negotiation Approaches

In an ideal case, the negotiation process lasts only for one round since the data consumer accepts an *offer* policy as it is published and the data provider issues the *agreement* policy accordingly. At most, a data provider provides the possibility to choose N out of M optional ODRL policy rules and the data consumer then selects those rules that match his demands. We call this approach a fully automated approach in which no further modification of the offered policies is accepted. On the other hand, in order to automate a human-like interactive approach with no restrictions, the parties need to build the *policy evaluation components* that interpret and define the policies like humans. This is a very complex task. Therefore, we propose to build a *policy evaluation component* that benefits from the extracted policy classes and evaluates the received policies, accordingly. Also, it generates a corresponding new policy with or without user inputs. As listed in Table 1, the interactive approach with limited modifications provides a platform that uses such a proposed *policy evaluation component* and allows the parties to negotiate over predefined, known negotia-

Algorithm 1: Transform ODRL Policy to a MYDATA Policy.

```
Input: ODRLPolicy op, bool provider-side
Output: MYDATAPolicy mp
MYDATAPolicy mp := NULL;
Mechanism mech := NULL;
function transformODRLtoMYDATA(op, provider-side)
mp := createMYDATAPolicy();
mp.pid := op.pid;
if provider-side = true then
    mp.solution := op.rules[0].assigner;
else
    mp.solution := op.rules[0].assignee;
end if
for all Rule rule in op.rules do
    mech := createMechanism();
    mech.event := rule.action;
    mech.conditions.add(createPIPCondition("target", "equal", rule.target));
    if rule.duty is not NULL then
        if rule.duty ≠ "anonymize" then
            mech.conditions.add(createPXP(rule.duty.action, rule.duty.refinement));
        else
            mech.conditions.add(createModifyFuntion(rule.duty.refinement));
        end if
    end if
    //The same for refinements
    if rule.constraints is not empty then
    for all Constraint rc in rule.constraints do
        if rc.leftOperand = "event" then
            mech.conditions.add(createCountCondition(rc.rightOperand, 1));
        else if rc.leftOperand = "count" then
            mech.conditions.add(createCountCondition(rule.action, rc.rightOperand));
        else if rc.leftOperand = "dateTime" then
            mech.conditions.add(createDateAndTimeCondition(rc.operator, rc.rightOperand));
        else
            mech.conditions.add(createPIPCondition(rc.leftOperand, rc.operator,
            rc.rightOperand));
        end if
    end for
    end if
    if rule.rule = "permission" then
        mech.decision := "allow";
    else if rule.rule = "prohibition" then
        mech.decision := "inhibit";
    end if
    mp.mechanisms.add(mech);
end for
return mp;
end function;
```

tion parameters. This approach does not support unlimited adjustments and modifications, however, the data provider and data consumer parties may collaborate when the human interaction is needed. For example, when a data consumer proposes a new time slot, the *policy evaluation component* detects the changes and can decide autonomously whether the request is accepted or not by comparing the proposed time slot with the defined demands (e.g., it can be demanded that the proposed time slot does not cross the lim-

Table 1: The policy negotiation approaches.

	Unlimited adjustments	Further policy evaluation	Applicable in IDS context
Human-like approach	✓	✓	×
Fully automated approach	×	×	✓
Interactive approach with limited modifications	×	✓	✓

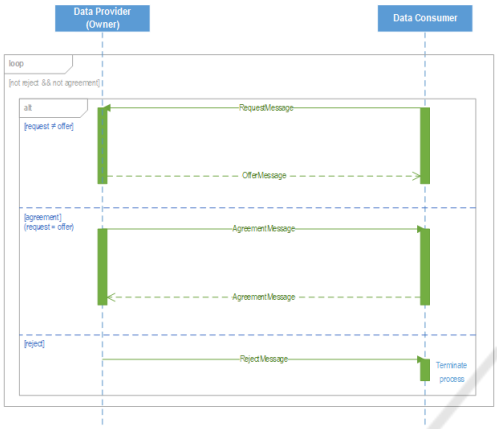

Figure 4: Policy Negotiation sequence diagram.

ited duration of one month). On the other hand, when a data consumer requests to use the data for different purposes than what has been offered, the approval of the detected request may need the data provider's manual intervention.

5.2 Policy Negotiation Parameters

We extracted the parameters that can be modified and thus negotiated during the policy negotiation process, with respect to the policy classes. To this end, we categorized the policy negotiation parameters into five groups: *decision*, *condition left operand*, *condition right operand*, *modification method* and *action*.

- **Decision:** In a first place, one can negotiate over a decision that is proposed in a Data Usage Control policy. Although, the request to change the decision of a policy (e.g., from prohibition to permission) is not frequently demanded, it still is a valid change. For example, a data provider provides an *offer* policy consisting of various rules, one of which inhibits the data consumer to print the data. In this case, it is worthwhile that the data consumer requests printing permission while accepting the other rules.
- **Condition Left Operand:** As described in Section 3.1, the stakeholders control the usage of data by restricting the usage to specific dates, time slots, systems, purposes, roles or users as well as on occurrences of specific events. In fact, one can negotiate about the left operands of these conditions. For example, a data provider may propose an *offer* policy that restricts the usage of data to marketing *purposes*. Here, a data consumer can request to restrict the data usage to marketing *systems* instead.
- **Condition Right Operand:** In most cases, the parties negotiate not to change a condition left operand, but to adjust the value, called right operands, that is assigned to it. The right operands and their corresponding value types must be certainly defined and known to the parties, since they define under which circumstance a condition is satisfied. Correspondingly, the *policy evaluation components* can detect and evaluate the proposed right operands and decide whether they accept it or not.
- **Modification Method:** Some policy classes describe that IDS use cases require data modification and data anonymization (at rest or in transit). The modification method (e.g., aggregate or delete sensitive information) is also a parameter that is negotiable between data provider and data consumer.
- **Action:** The policy action is another policy negotiation parameter. For example, a *policy evaluation component* on the data provider side may automatically accept a data consumer's request to print the data on paper rather than displaying it or to send a notification rather than sending the entire usage logs. The data provider may configure the *policy evaluation component* by setting the exchangeable actions that are equally acceptable. Using such a predefined table, the least intervention of the data provider is required to obtain an agreement.

Overall, the policy negotiation parameters may expand according to the updates on the policy classes.

6 CONCLUSIONS

In this paper, we used the policy classes that are extracted from the IDS requirements specification

as a basis for a systematic approach to create the technology-dependent policies. Since the companies' requirements evolve over time, the extracted policy classes must be updated or extended accordingly. The policy templates, policy transformation rules and the policy negotiation parameters will be updated as well and therefore, the IDS platform can claim to support users in their policy specification process. As a future work, we plan to implement our proposed policy negotiation platform and attach it to our policy editor.

ACKNOWLEDGEMENTS

This research has been funded by the Federal Ministry of Education and Research (BMBF) of Germany in the project "Architekturtopologien für Datensouveränität in Geschäftsökosystemen auf Basis des Industrial Data Space" (grant agreement number FKZ 01IS17031).

REFERENCES

Bennicke, M. et al. (2003). Towards automatic negotiation of privacy contracts for internet services. In *The 11th IEEE International Conference on Networks, 2003. ICON2003.*, pages 319–324. IEEE.

Bussard, L., Neven, G., and Preiss, F.-S. (2010). Downstream usage control. In *2010 IEEE International Symposium on Policies for Distributed Systems and Networks*, pages 22–29. IEEE.

Fraunhofer IESE (2018a). Mydata control technologies. https://www.mydata-control.de/.

Fraunhofer IESE (2018b). Mydata policy language documentation. version 3.2.69. https://developer.mydata-control.de/language/.

Fraunhofer IESE (2019). Odrl pap. http://odrl-pap.mydata-control.de/.

Iannella, R., Steidl, M., Myles, S., and Rodríguez-Doncel, V. (2018). Odrl vocabulary & expression 2.2. https://www.w3.org/TR/odrl-vocab/.

Iannella, R. and Villata, S. (2018). Odrl information model 2.2. https://www.w3.org/TR/odrl-model/.

Jung, C., Eitel, A., and Schwarz, R. (2014). Enhancing cloud security with context-aware usage control policies. In *GI-Jahrestagung*, pages 211–222.

Kelbert, F. and Pretschner, A. (2012). Towards a policy enforcement infrastructure for distributed usage control. In *Proceedings of the 17th ACM symposium on Access Control Models and Technologies*, pages 119–122. ACM.

Lorenzo, G. G. and Rodríguez-Doncel, V. (2018). Odrl editor. http://odrleditor.appspot.com/.

Narouei, M., Takabi, H., and Nielsen, R. (2018). Automatic extraction of access control policies from natural language documents. *IEEE Transactions on Dependable and Secure Computing*.

Otto, B., Auer, S., Cirullies, J., Jürjens, J., Menz, N., Schon, J., and Wenzel, S. (2016). White paper: Industrial data space. http://s.fhg.de/juA.

Otto, B. et al. (2018). Ids reference architecture model. industrial data space. version 2.0. http://s.fhg.de/2Qu.

Pauer, A., Nagel, L., Fedkenhauser, T., Fritzsche-Sterr, Y., and Resetko, A. (2018). Data exchange as a first step towards data economy. http://s.fhg.de/EAr.

Pretschner, A., Hilty, M., Basin, D., Schaefer, C., and Walter, T. (2008). Mechanisms for usage control. In *Proceedings of the 2008 ACM symposium on Information, computer and communications security*.

Pretschner, A. and Walter, T. (2008). Negotiation of usage control policies-simply the best? In *2008 Third International Conference on Availability, Reliability and Security*, pages 1135–1136. IEEE.

Rudolph, M., Moucha, C., and Feth, D. (2016). A framework for generating user-and domain-tailored security policy editors. In *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)*, pages 56–61. IEEE.