sustainability

MDPI

Article

Data Usage and Access Control in Industrial Data Spaces: Implementation Using FIWARE

Andres Munoz-Arcentales *, Sonsoles López-Pernas, Alejandro Pozo, Álvaro Alonso, Joaquín Salvachúa and Gabriel Huecas

Departamento de Ingeniería de Sistemas Telemáticos, ETSI Telecomunicación, Universidad Politécnica de Madrid, 28040 Madrid, Spain; sonsoles.lopez.pernas@upm.es (S.L.-P.); alejandro.pozo@upm.es (A.P.); alvaro.alonso@upm.es (Á.A.); joaquin.salvachua@upm.es (J.S.); gabriel.huecas@upm.es (G.H.)

* Correspondence: joseandres.munoz@upm.es

Received: 10 March 2020; Accepted: 3 May 2020; Published: 9 May 2020

Abstract: In recent years, a new business paradigm has emerged which revolves around effectively extracting value from data. In this scope, providing a secure ecosystem for data sharing that ensures data governance and traceability is of paramount importance as it holds the potential to create new applications and services. Protecting data goes beyond restricting who can access what resource (covered by identity and Access Control): it becomes necessary to control how data are treated once accessed, which is known as data Usage Control. Data Usage Control provides a common and trustful security framework to guarantee the compliance with data governance rules and responsible use of organizations' data by third-party entities, easing and ensuring secure data sharing in ecosystems such as Smart Cities and Industry 4.0. In this article, we present an implementation of a previously published architecture for enabling access and Usage Control in data-sharing ecosystems among multiple organizations using the FIWARE European open source platform. Additionally, we validate this implementation through a real use case in the food industry. We conclude that the proposed model, implemented using FIWARE components, provides a flexible and powerful architecture to manage Usage Control in data-sharing ecosystems.

Keywords: data economy; Industry 4.0; data governance; data usage control; data access control; IoT; shared data; FIWARE; International Data Spaces; UCON; usage policies; XACML

1. Introduction

Extracting value from data is one of the key aspects leading the development of applications and services, especially for Industry 4.0 [1]. Consequently, ensuring data governance and traceability becomes imperative to promote the exchange of data in this new business paradigm. Another crucial aspect of the data economy is interoperability [2], since sharing data between different stakeholders brings many new opportunities for all parties involved. In this scope, the use of trusted and secure platforms for sharing and processing personal and industrial data is essential for the creation of a data market and a data economy.

There is no doubt that big data analysis has played an important role in the fourth industrial revolution as it helps to determine a low cost strategy for companies to be more competitive and to identify how to increase their revenue and optimize their processes [3]. Furthermore, IoT has become one of the core elements in Industry 4.0 [4] as well, since it facilitates factories the tasks of speeding up their product development, achieving a more flexible production, and setting up more complex environments. Currently, smart factories are growing in number and they have the capability of manufacturing intelligent and customized products in a short period of time considering customers' preferences in real time [5]. In this regard, the streaming data gathered by IoT devices

have different features in comparison with traditional (batch) big data in terms of data generation, data interoperability, and data quality. Interpretation of big datasets from IoT is a challenge because the data sources are ubiquitous; the transmitted data are noisy, heterogeneous, and spatiotemporal dependent [6]. Thus, it is imperative to analyze these topics to determine what is the most suitable solution for establishing a cross-industry data-sharing ecosystem.

Regarding data sharing and in terms of security, two concepts play an important role: Access Control and Usage Control. Access Control constrains what a user can do directly, as well as what programs executing on behalf of the users are allowed to do [7], whereas Usage Control extends the traditional Access Control by enabling the control of a resource during and after the access to that resource has been granted [8]. In data sharing not only is it important to control who can access what resource (covered by identity and Access Control), but also how data are allowed to be used once accessed. This is of special relevance in publish/subscribe scenarios in which data are sent in real time to data consumers. In such cases, after providing the data, providers have no knowledge or control over how data are treated, leaving consumers the possibility to make unauthorized or outlawed actions over data that may have implications such as privacy or anonymity violation, which are some of the major concerns of the General Data Protection Regulation (GDPR) [9]. Access Control does not suffice to ensure that, once accessed, data are being used for the intended purpose. Thus, this measure needs to be extended to a more comprehensive model, such as Usage Control, in order to meet the industry's new requirements, guaranteeing that data consumers make appropriate use of data.

One of the foundations of data sharing environments is trust. Trust management concerns guaranteeing privacy, securing the exchange of data among different stakeholders, and ensuring that data are duly used once accessed. Standardizing trust systems fosters data transactions in the marketplace and eases compliance with the data governance framework defined by each organization. The International Data Spaces (IDS), formerly known as Industrial Data Spaces, Reference Architecture Model establishes a trust model to fulfill these requirements in the scope of Industry 4.0 and defines data Usage Control as one of the fundamental pillars to be addressed in the new digital revolution [10].

In the scope of Usage Control, the Usage Control model (UCON) stated by Sandhu and Park [8] was presented as a new framework for providing not only traditional data Access Control but also enabling the control of the data during and after the usage. The UCON model extends the traditional Access Control models, not only by introducing new concepts of mutability applicable to attributes of subjects and objects but also by providing a new way of guaranteeing the continuity of policy enforcement. Moreover, UCON presents a family of ABC models built around three decision factors: authorizations (A), obligations (B) and conditions (C), which can all be used for pre-decision and ongoing decisions. As a complementary part of the conceptualization of UCON, one of the formal model proposals of policies aligned with Usage Control is presented by [11], in which they attach two sets of predicates, namely provisions and obligations. On the one hand, provisions are concerned with the *pre-decision* phase (past and present), corresponding to the traditional Access Control. On the other hand, obligations deal with the rules applied to the future use of data and are related to the *ongoing decision* phase which, instead, is executed after the access is initiated and implements the continuity of control over the data. The new features introduced by UCON derives in several attempts to implement Usage Control. However, the development of a complete framework of data Usage Control applied to industrial data-sharing ecosystems is still an open issue.

This article relies on a comprehensive architecture for providing data access and Usage Control in industrial data-sharing ecosystems. This proposal incorporates the core concepts from the UCON model, the key aspects of the IDS Reference Architecture Model and the extended XACML (eXtensible Access Control Markup Language) Reference Architecture [12]. Moreover, the architecture is integrated into a complete identity and Access Control solution with support to delegated application-scoped security management [13,14]. Thanks to this integration, data providers can easily manage access and usage policies scoped to different application scenarios. The conceptual model of the architecture was

previously described in [15]. This article extends said work by providing an implementation of the proposed architecture and a validation with an original case study in the food industry.

The proposed architecture fulfills the main requirements [16] of Internet of Things (IoT) applications developed in shared data scenarios. It guarantees reliability and interoperability through the trusted and standardized environment established thanks to the IDS framework. Likewise, IoT applications are usually very demanding in terms of scalability. Thus, the data Usage Control architecture needs to meet this requirement. The extended XACML-based architecture separates each component based on its role, so each of these components can be scaled as needed. In terms of dynamism, the proposed architecture can easily apply predefined access and usage policies to the new dynamically added nodes, as it stores and manages those policies in a centralized way.

In this scenario, the FIWARE platform (FIWARE: The open source platform for our smart digital future, https://www.fiware.org/) seems particularly suitable for implementing a Usage Control architecture. FIWARE is an open initiative whose mission is to ease the development of new Smart Applications in multiple sectors by providing a set of components, known as Generic Enablers (GE), that enable the connection among IoT devices and Context Information Management and other services such as security or big data analysis. In a previous work [17], we proposed an implementation of the IDS architecture using FIWARE components focused on data brokering, identity and trust management, and the development of IDS Connectors. In the present work, we extend such implementation by adding the components that are needed to achieve Usage Control by data providers based on the architecture we propose. The newly added components introduced in this work are the Policy Translation Point (PTP), the Usage Policy Decision Point (uPDP) and the Policy Execution Point (PXP). A thorough explanation of every one of these components is presented in Section 4.

The article is structured as follows. Section 2 reviews the most relevant related work on data Usage Control. Section 3 presents a brief description of the architecture that we took as a reference for this implementation including some additional considerations for this proposal. In Section 4, an implementation of the proposed solution using FIWARE components is described. Section 5 presents a validation of said implementation by means of a use case in the food industry, which includes the results obtained from measuring the enforcement time when applying Usage Control policies. Lastly, Section 6 finishes with the conclusions of the article and an outlook on future work.

2. Related Work

According to [18–20], the adoption of big data and industrial IoT (IIoT) is rapidly growing in Industry 4.0. Previous works have provided industrial solutions that make use of this type of technologies such as recommendation and prediction systems, process optimization techniques, intelligent manufacturing, and AI applications [21,22]. In this regard, many open source big data tools have been developed to enable companies to process the vast amount of data that are generated by the IIoT. Several proposals present a general view of the big data architecture needed to manage industrial scenarios in which multiple sources of data are present [23,24]. The authors of [25] present a five-layer architecture for big data processing and analytics (BDPA): collection, storage, processing, analytics, and application. Since each layer involves different tasks, many open source tools have been developed to help overcoming some of them. Some examples include:

- **Collection**: Apache Kafka

  (Apache Kafka: https://kafka.apache.org/), Apache NiFi (Apache NiFi: https://nifi.apache.org/), Orion FIWARE Context Broker (FIWARE Orion: https://fiware-orion.readthedocs.io).
- **Storage**: CassandraDB (CassandraDB: http://cassandra.apache.org/), HDFS (HDFS: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html), MongoDB (MongoDB: https://www.mongodb.com/es).
- **Processing and analytics**: Apache Spark (Apache Spark: https://spark.apache.org/), Apache Flink (Apache Flink: https://flink.apache.org/), Apache Storm (Apache Storm: http://storm.apache.org/), FIWARE Cosmos (FIWARE Cosmos: https://fiware-cosmos.readthedocs.io).

- **Application**: Apache Zeppelin (Apache Zeppelin: https://zeppelin.apache.org/), Kibana (Kibana: https://www.elastic.co/es/kibana), GeoSpark (GeoSpark: https://github.com/DataSystemsLab/GeoSpark).

As stated in [6], big data processing in industry is different in several aspects from other scenarios. One of the major concerns in this sort of environments is the way in which data are collected, considering that data can derive from multiple sources. This issue has become more crucial in data ecosystems shared among multiple organizations, since each organization can process and store data in different ways and has different access and usage rights over data. Thus, we identify the need to use a common standard, compatible with the existing big data tools and IoT devices, that provides a standardized way to gather, extract, process and store data in industrial contexts. In this regard, NGSI-LD provides a simple and powerful open API published as an ETSI specification for the management of context information [26]. This specification promotes the adoption of a standard way to manage data in the whole industrial data processing pipeline. However, the use of a common standard makes it imperative to use big data tools that can work with data in this format. To address this issue, we include the FIWARE GEs in the industry's BDPA pipeline since they comprise a set of libraries, connectors, and protocols on top of the most widely used big data frameworks provided by the Apache Community, providing these components with full support for the NGSI-LD standard. With the adoption of this approach, any organization can not only perform big data operations more easily using a common standard, but also mitigate additional needs such as enforcing data Usage Control and securing data exchange.

As far as data Usage Control is concerned, most of the proposals in the literature take the UCON model [8] as a starting point for the development of their solutions. However, depending on the field of application, these works take different approaches. For instance, Russello and Dulay presented xDUCON [27], a cross-domain Usage Control proposal for coordinating and enforcing Usage Control policies across different collaborating organizations. In said framework, a cross-domain data space instance is shared among the organizations to be used as a local enforcement point of the control policies. As a result, the coordination of the enforcement policies is easier to specify since it is not necessary to include details of the receiving organization structure.

A posterior paper was presented by the same authors [28]. As a complementary part of the xDUCON framework, they defined cross-domain policies, which are capable of dealing with the mutability issues of the UCON model and providing a fine-grained decision mechanism that can be captured by the defined policies. The xDUCON framework provides a general perspective for providing capabilities of policy enforcement and specification. Furthermore, Di Cerbo et al. present a solution for avoiding security risks and providing a mechanism for allowing the data owners to keep the data under their control. They present [29] a solution that allows the provision of a secure data sharing across the cloud and mobile engines. This is achieved by relying the enforcing mechanism and rules definition on Policy Definition Languages (PSLs) like XACML and an extended version of PPL (PrimeLife Policy Language) [30]. However, these works fall short of providing an architecture that not only covers the enforcement and definition mechanisms for access and Usage Control but also provides a full description of the whole process that data Usage Control involves.

Likewise, Lazouski et al. use the principles presented in [31] for providing a Usage Control solution mainly focused on cloud systems applications. The conducted research is based on the UCON model and the OASIS XACML standard to regulate the usage of cloud resources [32]. This proposal was validated by implementing the authorization system integrated with OpenNebula (OpenNebula: https://opennebula.org). More comprehensive research was presented by Wu et al. [33], in which data Usage Control is enforced in industrial Wireless Sensor Networks (WSN). Not only do they provide cross-domain fine-grained Access Control, but they also use fuzzy clustering to analyze industrial sensing data. This work uses a set of simulations for verifying the suitability of the overhead time and the effectiveness of the proposed model. A comparable proposal was presented by Marra et al. [34]. They used the core concepts of UCON model and the XACML reference architecture

in order to implement a Java application for providing Usage Control operations over IoT devices. They developed a case study in which they evaluate the performance of the IoT devices and determined the feasibility of the system by implementing their proposal on real devices.

Recent works provide some interesting solutions for data Usage Control based on the XACML reference architecture proposal. For instance, Barsocchi et al. use GLIMPSE [35], a flexible monitoring infrastructure for performing Usage Control on operations over sensors in a smart home. They demonstrate the feasibility of carrying out Usage Control in this type of environments. In addition, in said study, the authors provide a low cost, easy to install, user-friendly, dynamic and flexible infrastructure, capable of performing run-time resource management using control rules [36]. Similarly, Gkioulos et al. present a model that can integrate access and Usage Control mechanisms for dealing with the distributiveness and heterogeneity of systems like IoT and online banking. At the same time they bring several improvements regarding resilience on active attacks, policy writing simplification, run-time efficiency and scalability [37] . Another proposal by Martinelli et al. presents a framework for applying QoS in a network in a Smart Building environment. They combine UCON and SDN (Software Defined Networks) in order to enforce a set of management, security, and safety policies aimed at ensuring the appropriate QoS for the provided services according to both the tenants' Service Level Agreements (SLAs) and the current context [38]. Lastly, an approach presented by Milan Petković et al. in [39] uses the business model of an organization to detect privacy infringements and to verify that data have been processed only for the intended purpose. The work presents a strong point in formal specification using Calculus of Orchestration of Web Services (COWS). These proposals show the growing interest of the scientific and industrial community in exploiting all the capabilities of Usage Control, and also demonstrate its application in specific scenarios (IoT, Cloud, etc.). Nevertheless, these overtures do not cover topics like cross-domain data exchange, data governance and trust environments, highlighting the need to deal with these topics before using this type of architecture in industrial data-sharing ecosystems. Moreover, in Industry 4.0, data sharing between multiple organizations is a key factor to be considered. Thus, guaranteeing the compliance with data governance rules and the responsible use of organizations' data by third-party entities is one of the requirements that needs to be addressed. Therefore, the need to generate a more flexible framework, capable of adapting to these mixed data ecosystems, is identified.

Moreover, the Data Privacy Directive 95/46/EC [40], currently replaced by the GDPR, played an important role in data protection. In this regard, many Access Control solutions are currently presented for protecting personal data. For instance, Bartolini et al. proposed a systematic approach for authoring Access Control policies that are aligned with GDPR provisions. They present a methodology for generating templates from the GDPR text and identifying if a GDPR article can be defined as an Access Control policy. This is achieved by matching actual attributes gathered from the legal use cases and translating the resulting policies into a given formalism or language [41] in order to comply with GDPR's principle of "data protection by design and by default" [42]. In the same line, Calabró et al. conducted a preliminary study for integrating Access Control and business processes for GDPR compliance. The main goal of said study was to extend the currently adopted Access Control mechanisms to enforce GDPR compliance during business activities of data management and analysis [43]. Nevertheless, although these works provide a first step towards a formal definition of an Access Control solution based on GDPR, they do not cover the Usage Control of the data once access is granted, opening an issue that needs to be addressed. Another interesting proposal was presented by Arfelt et al. who identifies formalizable GDPR articles and, by using Metric First-order Temporal Logic (MFOTL), formalizes and monitors the articles in which controllers, processors, or data subjects are required to take specific and observable actions [44]. Moreover, the policies generated with the previous process are deployed over MONPOLY, a monitoring tool for compliance checking [45]. Although previous works solve the problems of GDPR formalization and monitoring, these proposals do not consider other factors such as data governance and trust that affect data sharing in cross-domain industries.

Prior works have sought Access Control solutions based on Blockchain for distributed environments [46–48]. In particular, the authors of [46] describe how contracts can be deployed in the ledger to perform Usage Control following GDPR provisions and avoiding central entities in the authorization and authentication processes. Overall, the cited studies outline that relying on Blockchain provides transparency and trust solutions for Access Control but it may impose scalability limitations for real-time scenarios.

As can be seen, much of the research up to now has been descriptive in nature. Table 1 presents a comparison of the Usage Control architectures found in the literature. Most solutions rely only on ABAC (Attribute Based Access Control) for Access Control and just a few proposals supplement it with RBAC (Role Based Access Control) or IBAC (Identity Based Access Control), which play a crucial role in data-sharing scenarios in which it is necessary to provide particular access and usage permissions to different stakeholders. In addition, the existing Usage Control solutions focus on policy infringement detection. Only about half of the works studied provide remediation capabilities for policy violation, rather than just detection. Remediation actions are essential to automatize the enforcement of consequences for policy noncompliance rather than issuing warnings or notifications and leaving it up to the wrongdoer to redress the situation. In order to univocally define obligations, prohibitions and permissions, and the consequences for noncompliance, it is convenient to use a Policy Specification Language (PSL). Most works found in the literature rely on XACML or U-XACML as their PSL, although some of them do not use a specific language or define one of their own. Another important aspect is the support for multi-actor architectures. About half of the works studied provide support for several actors involved in the data-sharing process, whereas the rest of them focus on one-on-one exchanges. The former is a key requirement for scenarios in which multiple stakeholders share data with one another. Furthermore, it is worth mentioning that most of the studies analyzed in this section fail to provide a data Usage Control solution that is independent of the context in which data are generated (WSN, IoT, SDNs, etc.), with only a couple of them being domain-agnostic. Lastly, far too little attention has been paid to the topic of trust environments as a pillar for providing capabilities for secure and trusted data exchange and sharing between multiple organizations.

Table 1. Comparison of usage control architectures

References	Access Control	Remediation	PSL	Multi-Actor	Application Domain
[27]	ABAC		Cross-Domain Policy (xDPolicy)	✓	Generic
[29,30]	ABAC		PPL,XACML	✓	Cloud, mobile
[31,32]	ABAC, RBAC, IBAC		U-XACML	✓	Cloud
[33]	RBAC				Industrial WSNs
[34]	ABAC	✓	U-XACML		IoT
[36]	ABAC,RBAC	✓	Drools Rule Language (DRL)	✓	IoT
[37]	ABAC	✓	U-XACML		IoT
[38]	ABAC,RBAC				SDN
[39]	ABAC	✓	ad-hoc		Generic

In light of this information, it becomes apparent that previous works have failed to provide a standard solution for achieving data Usage Control in data-sharing ecosystems. None of the works reviewed provides advanced Access Control and Usage Control capabilities in architectures with several agents involved, while supporting an expressive policy definition language that allows the definition of obligations, prohibitions and permissions and providing remediation functionalities in

the event of policy infringement. This research contributes to fill the gap in the existing literature by providing a generic multi-actor architecture to achieve advanced data access and usage control capabilities in real-time data-sharing scenarios. Our proposal incorporates the core concepts from the UCON model, the key aspects of the IDS Reference Architecture Model and the extended XACML Reference Architecture, and relies on ODRL as its PSL. In addition, we provide an implementation using one of the core CEF (Connecting Europe Facility) building blocks and FIWARE GEs for providing a reference framework fully adapted to the requirements of Industry 4.0.

3. Proposed Solution

This section presents an overview of the design principles that we have considered for designing this proposal. We also summarize the resulting architecture as well as a workflow that illustrates the interaction between the described components. Details about both the principles and the architecture can be found in our previous work [15].

3.1. Design Principles

In recent years, industries have experienced an increased need to exchange data among them. Thus, data protection has become a priority. In view of this necessity, the IDS, which is in close contact with the industry, has identified the main requirements that need to be fulfilled in order to address data sharing when multiple organizations are involved. We have taken the IDS guidelines included in [10] as a starting point and added additional considerations from the literature [17] to concoct the set of design principles in which we have based our proposed architecture for providing access and Usage Control in industrial contexts:

- *Trust*. IDS Connectors provide a trusted environment that enables the achievement of data Usage Control [10].
- *Interoperability*. Standardization of protocols is crucial to ensure the understanding between all the components involved in the architecture, for managing both Usage Control and identity.
- *Governance*. Emerging data-centered businesses need to define data governance programs to exploit data in a cost-effective manner [49]. Data sharing should comply with the data governance rules defined by each of the organizations involved. In this scope, providing ways to respect and protect the data of all the parties involved is one of the main requirements that data Usage Control must fulfill. Thus, data providers must have access to monitoring and configuration tools that allow them to control what becomes of their data. Nevertheless, as pointed out by [50], in collaborative systems, resources can be administered by multiple data owners. Due to this fact, the aspects of data governance model, policy composition and conflict resolution need to be addressed. In this context, the concept of "data governance model" defines the authority that entities have over a resource; "policy composition" describes how the authorization requirements authored by multiple entities are combined or reconciled to regulate the access to a resource, and "conflict resolution" indicates the method used to resolve policy conflicts in order to obtain a conclusive decision [50]. In this regard, the preliminary version of this proposed architecture takes the work presented by Mahmudlu et al. as a reference, in which they define multiple ownership, authoritative and predefined mechanisms for addressing the main aspects of governance model, policy combination and conflict resolution respectively [51].
- *Performance*. The accomplishment data Usage Control policies can be only ensured if reaction to policies violation is quick and efficient.
- *Flexibility*. As many data-sharing scenarios and use cases are contemplated, the solution must be adaptable to the specific requirements of such scenarios.

3.2. Agents Involved

According to the International Data Spaces Association and IDS Reference Architecture presented by Fraunhofer [10] four agents have to be provided in every system that considers data sharing: Data

Owner (DO), Data Provider (DP), Data Consumer (DC) and Data User (DU). However, it is very frequent to assume a two actors model in which the DP and the DO play the same role as well as the DC and the DU. Taking into account this assumption, the DP is the organization or user who is the proprietary of the data and decides which data are available for sharing. Additionally, the DP defines the usage and Access Control policies applicable to the data that the DC can consume. On the other hand, the DC represent every entity that has the legal rights to use the data provided by a DP according to the previously defined Usage Control policies.

Besides these two actors, the GDPR, defines an additional component named Data Controller (DCr) [52]. According to the DCr definition, a third actor needs to be included to cover the IDS reference architecture and the one proposed by the GDPR. In this case, the DCr and the DP are responsible for guaranteeing the protection of data owners' rights and for providing access to data, respectively. Nevertheless, we consider that a model capable of being GDPR-compliant needs to contemplate some other factors, as stated in [43]), that are out of the scope of this paper. For instance, one of the additional factors that should be taken into account is the ability to write GDPR articles formally as algebraic expressions to transform legal concepts into rules. Also, it is necessary to provide a formal extension of a PSL to explicitly manage GDPR principles of consent and purpose limitation. Lastly, it would be useful to include tools for authoring and enforcing GDPR-based policies. Thus, we concentrate our efforts on presenting a preliminary version of data Usage Control with a two-actor model (DP-DC), mainly focused on data-sharing ecosystems in Industry 4.0. However, below we present an alternative proposal architecture including the three-actor model.

Finally, we also consider the Identity Provider (IdP) as an actor presented in the IDS Reference Architecture: The IdP verifies the authenticity of all the actors involved in the architecture and also provides all the characteristics related to identity management. These include actor registration, authentication, password management and the option of grouping actors in organizations to manage them under identical conditions. In scenarios in which a single DP shares data with one or more DCs, the DP can integrate its own IdP, since the identities of other DPs do not need to be validated. Said configuration would lead to an even further simplification of the two-actor model.

3.3. Architecture and Workflow

Building on the design principles established and the simplified agent model identified for data processing scenarios (DC, DP and IdP), we have presented an architecture for providing data usage and Access Control in shared ecosystems. Figure 1 shows the proposed two-actor architecture, whereas Figure 2 presents the aforementioned alternative architecture including the DCr as a separate agent.

On the other hand, Figures 3 and 4 show the workflow of the two main scenarios of data sharing: (1) DP defines the policies that apply to the shared data and (2) DC preforms operations to the received data.

The DP uses the Policy Administration Point (PAP) to define access and Usage Control policies. Usage policies are defined using an Open Digital Rights Language (ODRL) extension materialized by the W3C [53] and translated by the Policy Translation Point (PTP) to a program that runs on the usage Policy Decision Point (uPDP) and that is updated every time policies are modified by the DP. Once both access and usage policies are defined, the data in the Data Infrastructure (both real-time and stored) can be made available in the Shared Data Space (SDS).

When the DC requests access to data available in the SDS (to save them in its Storage System or to process them using a Processing Engine), the Policy Enforcement Point (PEP) checks with the access Policy Decision Point (aPDP) if the DC has the necessary access permissions to make the subscription effective. If the result is positive, the DC starts receiving data from the SDS and processing them performing the desired operations. As a result of such operations, traces are generated and sent to the PEP that after validating an authentication token previously generated by the IdP, redirects them to the uPDP. The uPDP checks the usage policies and in case of noncompliance delegates the responsibility

of enforcing the established action for policy noncompliance to the Policy Execution Point (PXP) by sending the corresponding control signal.

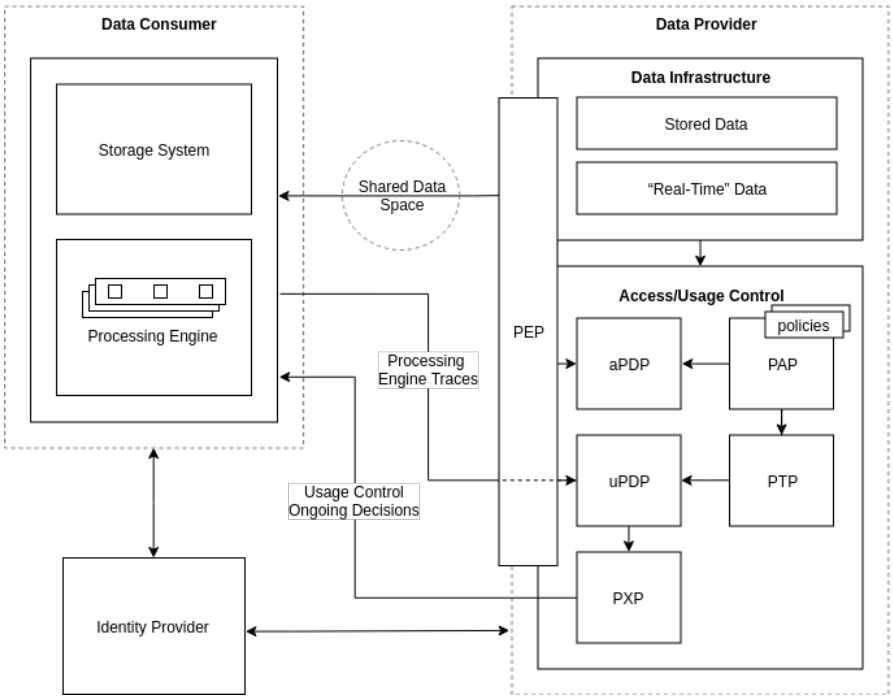

Figure 1. Data Usage Control architecture.

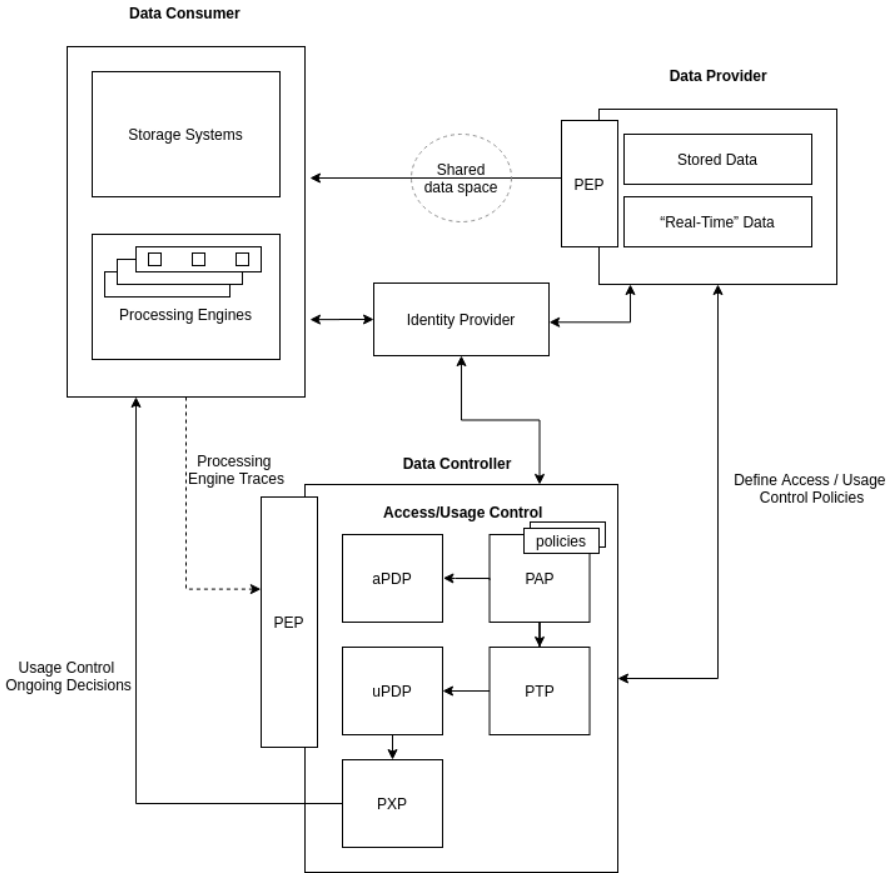

Figure 2. Data Usage Control with three actors architecture.

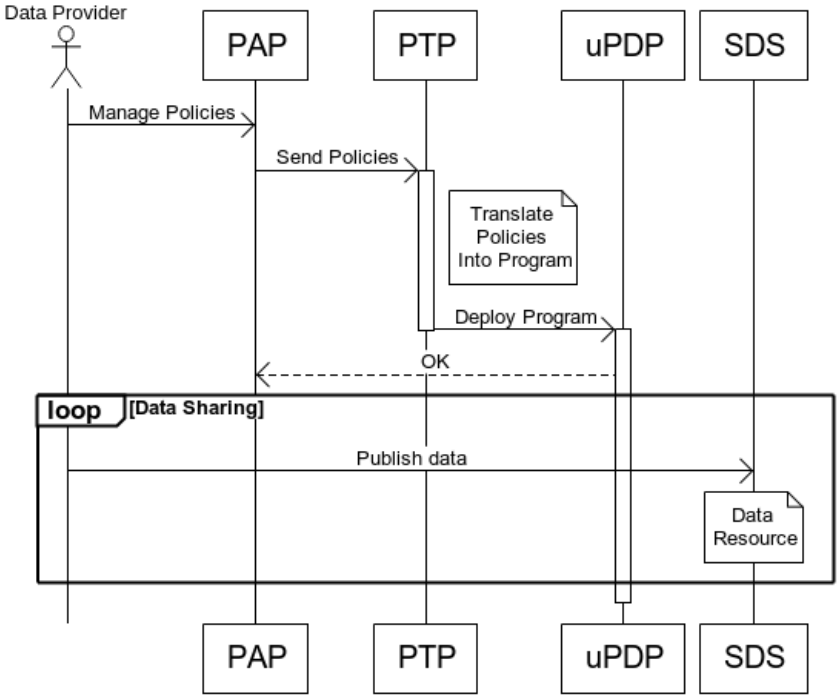

Figure 3. Workflow for the DP.

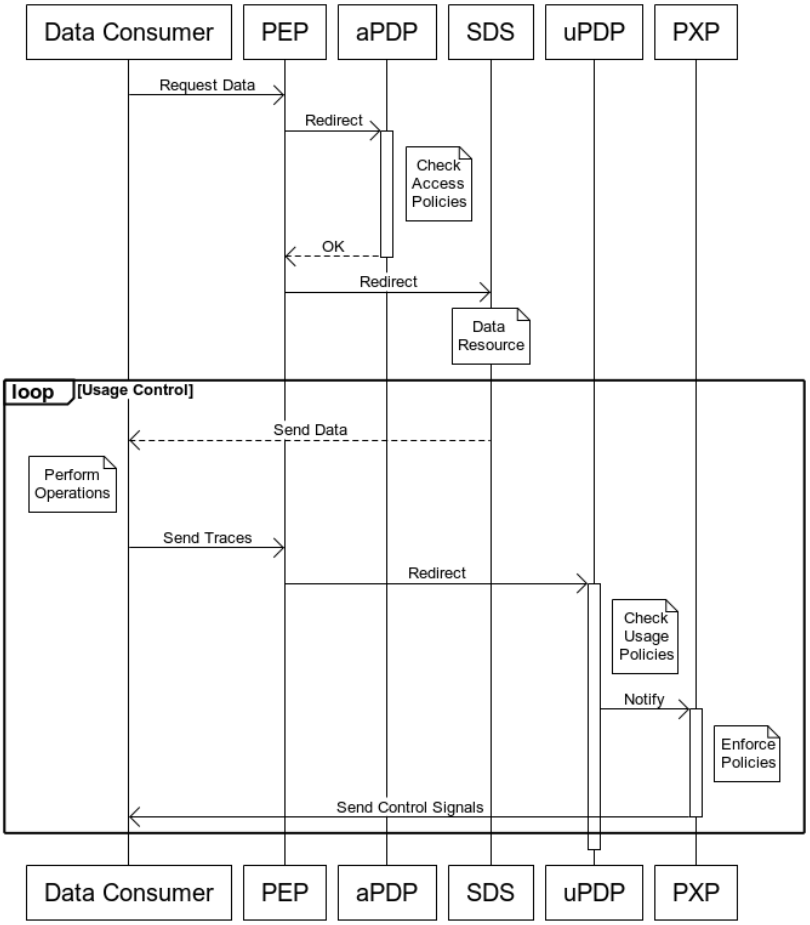

Figure 4. Workflow for the DC.

One concern identified in this proposal, is that the logs generated when performing operations on data could be easily manipulated due to the fact that log generation takes place outside the scope of the DP. Nevertheless, the reference architecture used in this work relies on the guidelines of the IDS connectors, which determine that all the connectors involved in a data exchange must run a trusted (certified) software stack. Worded differently, IDS Connectors require a certification from the IDS Certification Body to establish trust among all participants.

Moreover, the IDS guidelines also establish that any communication between connectors from different organizations should be encrypted and integrity protected. Thus, by including the DP and DC inside IDS connectors, each DP is capable of ensuring that their data are handled by the Connector of the DC according to the usage policies specified, or else the data will not be sent [10].

4. Implementation Using FIWARE

This section presents an implementation of the proposed architecture using the generic enablers (GEs) provided by FIWARE and other open source tools. Specifically, the GEs used in this implementation of the data usage architecture are the following:

- **Keyrock** The Keyrock GE (FIWARE Keyrock: https://fiware-idm.readthedocs.io) is responsible for Identity Management. Using Keyrock enables OAuth 2.0-based authentication and authorization security to services and applications, as described in [13,14]. In the context of this implementation, Keyrock plays the role of **IdP**, manages authorization policies (**PAP**) and decides which DCs can access which resources in the data infrastructure (**aPDP**). Therefore, DPs and DCs perform the authentication process relying on Keyrock. Guaranteeing the unequivocal identification of all the agents involved in the data usage architecture is mandatory to ensure a secure way of providing or consuming data. By using Keyrock, DPs can create authorization policies to constrain DCs' access to the data infrastructure.
- **Wilma**: The Wilma GE (FIWARE Wilma: https://fiware-pep-proxy.readthedocs.io) brings support of proxy functions within OAuth 2.0-based authentication schemas. It also implements **PEP** functions within an XACML-based Access Control schema [12]. In the scope of this implementation, two Wilma instances are needed. One Wilma instance is in charge of enforcing access policies over requests sent to the data infrastructure [17]. When a DC is authenticated through Keyrock, an OAuth 2.0 token is generated, which must be included in every request sent to the DP's data infrastructure. Wilma intercepts requests and asks Keyrock to validate the token, verifying the DC's identity. Since Keyrock also acts as the **aPDP**, it checks the DC's access authorization policies. In case that the DC's request complies with the established policies, Wilma grants access to the requested resource. With regard to data Usage Control, a second Wilma instance is needed as a **PEP** proxy to authenticate the traces sent from the DC's processing engine to the uPDP.
- **AuthZForce**: The AuthZForce GE (FIWARE AuthZForce: https://authzforce-ce-fiware.readthedocs.io) brings additional support to **aPDP/PAP** functions within an Access Control schema based on the XACML standard. It has not been included in the present implementation, but it could be used to create more advanced fine-grained authorization policies and to make decisions over requests received from PEPs.
- **Orion**: The Context Broker (Orion) GE (FIWARE Orion: https://fiware-orion.readthedocs.io) manages the entire lifecycle of context information including updates, queries, registrations and subscriptions. The Context Broker offers the FIWARE NGSI (Next Generation Service Interface) [54] APIs and associated information model (entity, attribute, metadata) as the main interface for sharing data among stakeholders. In addition to being the centerpiece of any platform "powered by FIWARE", the Context Broker has been recognized as a CEF Building Block, which is one step forward on its path towards becoming a global standard for large scale contextual information management [55] . In the context of this implementation, it constitutes the DP's **data infrastructure** and **SDS**, which enables the sharing of data between the DC and the DP in a secure

way. In other words, the DP makes use of the NGSI API provided by the Orion Context Broker in order to publish or expose the data they have to offer, whereas DCs retrieve or subscribe to said data.

- **Cosmos**: The Cosmos GE (FIWARE Cosmos: https://fiware-cosmos-flink.readthedocs.io) provides an interface for integrating Apache Flink and Apache Spark with the rest of the components in the FIWARE Ecosystem. Over recent years, Apache Flink and Apache Spark have established themselves as the most popular open source data processing frameworks. Both provide a wide assortment of resources for processing data both in streaming and batch modes. Since Apache Spark and Apache Flink provide similar functionalities, an implementation with only one of these frameworks is enough to validate the proposed architecture. Therefore, for the implementation presented in this work, we have relied on the Apache Flink as the **processing engine** on the DC's side, in charge of performing operations on the DP's data.
- **Draco**: The Draco GE (FIWARE Draco: https://fiware-draco.readthedocs.io) is aimed at providing storage of historical context data, allowing the reception of data events and dynamically recording them with a predefined structure in several data storage systems. In the scope of this implementation, Draco is proposed as the building block for providing the **storage system** in the DC's infrastructure.

The aforementioned FIWARE GEs provide all the features needed to implement the components on the DC's side (processing engine and data storage), the Access Control components (PAP, PEP, and aPDP), the SDS, and the IdP. As the FIWARE catalogue lacks any GEs that aid in the implementation of Usage Control capabilities, we have developed several ad-hoc components for this purpose. The Technical Steering Committee of FIWARE has shown interest from the conceptualization of this proposal to its materialization since it covers the key integration aspects of cross-industry data exchange. As some of the authors of this work are part of such committee, it is directly connected with the design of our solution. The PTP, uPDP and PXP components are planned to be included as a new FIWARE GE in the near future:

- The **PTP** is a piece of software written in Python in charge of translating the ODRL usage policies defined through Keyrock into a Complex Event Processing (CEP) program using the Flink CEP Scala API. Every time the usage policies that apply to a certain DC are modified by the DP, a new program is generated by the PTP containing a CEP rule for each policy. This program is then compiled, packaged, and sent to the uPDP.
- The **uPDP** is an Apache Flink computation cluster that runs all the CEP programs generated by the PTP: one for each DC. These programs take advantage of the CEP capabilities of Apache Flink to verify whether the DC complies with the policies defined by the DP or not. This is done by analyzing the traces generated by the processing engine on the DC's side (Apache Flink in this case) which are sent to the uPDP. In the event of noncompliance, the PXP is notified.
- The **PXP** is a piece of software that is notified each time the uPDP detects policy noncompliance. It is written in Scala and attached to each program that runs on the uPDP. The PXP enforces the control signal established by the DP for the unfulfilled policy. For instance, in order to stop a DC from receiving data as a punishment for policy noncompliance, the control signal sent by the PXP is an unsubscription request to Orion. If the DC is, in turn, processing data in an incorrect manner, one way to punish this policy violation would be to send a control signal that kills the processing job on the DC's side. These are the control signals that have been implemented so far, but the goal is to extend the capabilities of the PXP to support custom control signals written by the DP.

Figure 5 shows the data usage architecture proposed using the aforementioned FIWARE GEs and ad-hoc components developed, as well as the workflow mechanism presented in Section 3.3. Instead of deploying the IdP as an external actor (as proposed in Figure 1), in Figure 5 we include the IdP as a part of the DP, since the IdP is provided by the Keyrock GE, which also includes the PAP and aPDP. However, a three-actor configuration like the one proposed in Figure 2 would also be feasible by

deploying Keyrock separately, since Keyrock supports a multi-tenant configuration in which each DP would be mapped to a specific application. In such case, the Access Control permissions and usage policies that apply to a specific DP would be defined and validated in the scope of the corresponding Keyrock application. Details about the application-scoped Access Control management of Keyrock can be found in [13].

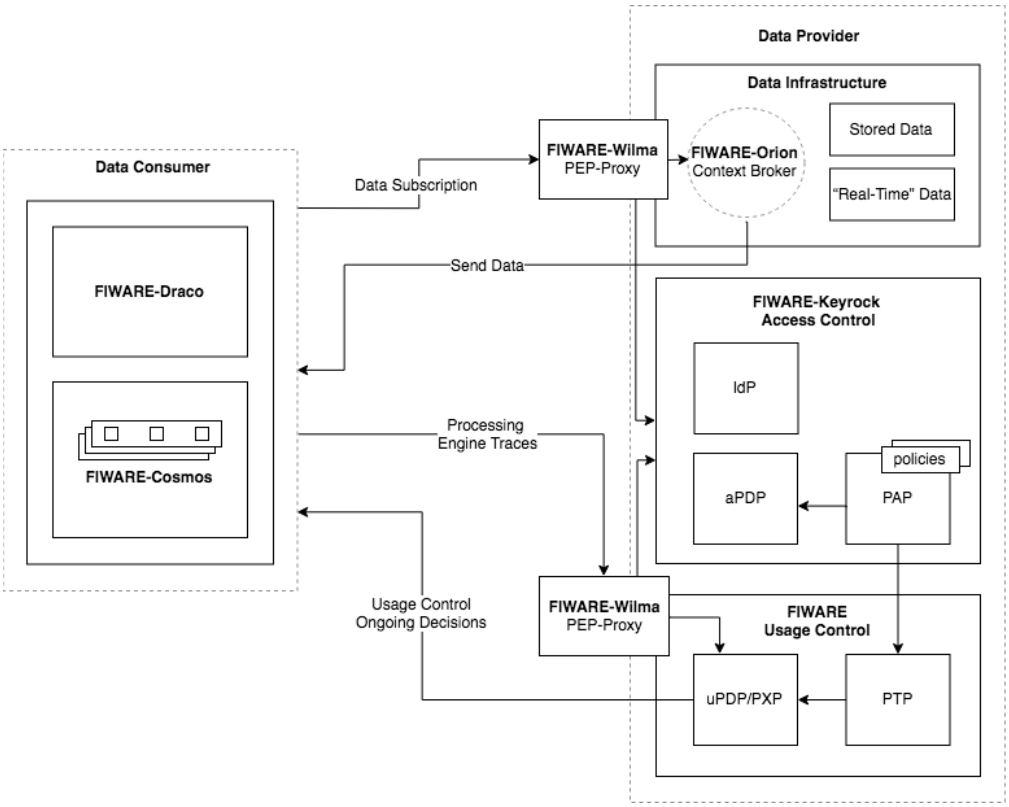

Figure 5. Data Usage Control architecture using FIWARE.

The operation flow during one usage decision process is defined as follows:

- The DC sends a subscription request to the Orion Context Broker to retrieve data from the DP.
- The subscription request is intercepted by the access PEP Proxy and validated by the IdP and the aPDP by checking whether the token containing the user information is valid an if the user has the right to access the requested resource.
- Once the subscription is done, the DC starts receiving data from the Orion Context Broker at the processing engine. The traces generated by the program containing all the operations performed on data are sent to the uPDP, verifying the DC's identity through the usage PEP Proxy. Moreover, this instance of the PEP Proxy is in charge of redirecting the traces to each specific uPDP CEP program. When translating the defined ODRL policies for a DC, the PTP generates a new CEP program and maps the port where it runs to the DC. Thus, when receiving the traces and after verifying the DC's identity, the PEP Proxy knows the port in which the corresponding CEP program is running and can redirect the traces to it. The uPDP then verifies that the DC complies with all the policies defined through the PAP. Otherwise, the uPDP notifies the PXP, who sends the corresponding control signal.

To ensure integrity, confidentiality and authenticity in the exchange of traces between DC and DP, we take advantage of the facilities that IDS Trusted Connectors provide to avoid eavesdropping, manipulation and impersonation [17]. The IDS defines two layers of security with regards to communication between Connectors: point-to-point encryption using an encrypted tunnel and end-to-end authenticity and authorization. The DC sends the traces generated by the processing

engine to the DP using HTTP requests over the Internet or through a Virtual Private Network (VPN), depending on the specific scenario. Regardless, HTTPS (the secure version of the HTTP protocol) is used in both cases. HTTPS [56] makes use of an added encryption layer of SSL/TLS to protect the HTTP traffic. Therefore, the point-to-point encryption is taken care of by this protocol. On the other hand, end-to-end authenticity and authorization are covered using the OAuth 2.0 protocol. As explained above, the DC includes an Authorization Header with the OAuth 2.0 token previously created by the IdP in the HTTPS requests containing the traces. As the PEP Proxy intercepts such requests before sending them to the uPDP, the identity and permissions of the DC are validated with the IdP and the aPDP to ensure authenticity and authorization respectively.

Lastly, verifying the correct timestamping of the traces received is also crucial to avoid replay-attacks [57]. Replay-attacks consist of resending an already sent request (trace) (maybe repeatedly). In our proposal, timestamping verification is also delegated to the use of OAuth 2.0 by means of the inclusion of timestamps and nonces (nombers used once) in each one of the traces generated by the DC. Adopting this mechanism, we can ensure that even if an attacker tries to replay the trace, this request will be denied by the PEP Proxy because it is not possible to neither change the timestamp nor the nonce used, since these values are also used in the signature (changing them would invalidate the signature).

Besides the secure interchange of traces between DC and DP, privacy, non-repudiation and integrity must be ensured in the whole data-sharing process, including the subscription and publication requests between the Orion Context Broker (on the DP's side) and the DC components. These requests are also protected thanks to the security features of IDS Trusted Connectors explained above. Thus, using encrypted requests, attackers cannot access shared data by brute force and, since all the requests are signed, the identity of the actors cannot be impersonated.

5. Validation: A Case Study in the Food Industry

To validate the proposed architecture and implementation using FIWARE, a case study has been developed in the food industry. The components presented in the implementation section have been deployed to perform the policy definition and enforcement in a shared data ecosystem. The main goal of this case study is to perform access and Usage Control over industrial data.

5.1. Scenario Overview

The scenario is composed by two actors: a food company (DP) and a marketing company (DC). The former generates a great amount of data daily every time a client makes a purchase at one of their grocery stores, which are later used for internal big data analysis. One data record is generated for each purchase, which consists of the client id, the payment method, and the list of products purchased, including the product name, price and quantity. The board of the company realizes that if they were to share these data, they would allow other businesses to find new ways of extracting value from them, thus creating another source of revenue for the company. A marketing company is interested in the food company's real-time data to identify trends and carry out instantaneous special offers that take these into account. In order for the marketing company to be able to make this analysis, the food company must provide a real-time channel to make their data available to them. However, the food company wants to keep the marketing company from making an incorrect use of the data that would jeopardize customers' privacy. For the sake of data protection, a set of Usage Control policies are defined to enforce some constraints over the shared data. In addition, since the communication channel between the DP and the DC is in real time, it is impossible to know a priori the number of data events there are going to be generated.

In this scenario, the DC deploys a Flink cluster for performing all the data processing operations. On the other hand, the DP deploys all the components showed on the right side of Figure 5 (i.e., the Orion Context Broker, Keyrock, Wilma, and the proposed Usage Control components), including the

data generated and published by the cash registers on the Context Broker as part of the data infrastructure stakeholder.

To simulate the grocery store data, we have extracted data from real purchases from an open dataset released by a very popular French grocery store chain. We converted these data into a stream of real-time notifications by triggering purchases periodically (in periods ranging from 25 ms to 5 s). Each notification represents a single purchase and contains a timestamp, the payment method used, and a comprehensive list of the items purchased, including, for each item, the quantity and the price. As can be seen, very fine-grained information related to consumer habits can be extracted from these data. Also, it becomes apparent that the more notifications the marketing company receives, the more accurate their offers will be for the stores' customers. It would be interesting to be capable of limiting the throughput of data events to implement different monetization strategies.

5.2. Policy Specification

In the proposed scenario, the DP defines two main policies regarding data usage that apply to any external DC. The natural language definition for these policies is:

- **Policy A**: The DC shall NOT save the data without aggregating them every 15 min first or else the processing job will be terminated
- **Policy B**: The DC shall NOT receive more than 200 notifications from the Context Broker in 1 min or else the subscription to the entity will be deleted

Policy A tackles one of the main concerns in data-sharing scenarios, which is anonymization. For instance, individual purchases could be cross-referenced with credit-card statements inferring the identity of the client and his/her consumer habits. By requiring the DC to aggregate data, the individual attribute values in each notification are combined into a single value at least every 15 min (e.g., by computing the mean, the maximum, the sum, etc.), thus guaranteeing that individual records are not saved. If, for instance, the DC tries to print the data or send them somewhere else upon receipt, this policy will be violated since the entirety of the data would be transferred away from the scope of data Usage Control without anonymizing them first. In the event of policy violation, the job will be immediately terminated by sending a signal to the DC's Flink cluster manager. Regarding Policy B, in scenarios involving large amounts of data, it is often useful to establish a limit in the throughput of data that is shared (i.e., amount of notifications in a given time). As mentioned, one possible application would be establishing different monetization strategies based on the maximum throughput of notifications allowed. In this case, the limit is set to 200 notifications per minute. In the event of noncompliance, the subscription to the entity for which the limit was surpassed will be removed.

As mentioned, in order to take full advantage of all the capabilities of data Usage Control, usage policies must be defined by using a policy specification language and, although ODRL provides a powerful interface to define these [58], in the future, it will be necessary to develop new vocabularies and ontologies for tackling some currently uncovered cases. However, in this case study we include a first approximation of the use of ODRL to declare the two policies that have been presented in natural language. In Listing 1, we present the ODRL definition of policies A and B. ODRL defines three ways to declare policy rules: "permissions", "obligations" and "prohibitions", providing different options to express a policy. We use "obligations" combined with "constraints" and "consequences" for defining our two policies. In each obligation, there is a "target", which refers to a resource that is subject to a rule, and an "action", which is the operation that is forced to be perform on the target as part of the obligation. Actions can be limited by "constraints", which can be temporal, spatial, amount-based, etc. In addition, "consequences" allow definition of what happens in case of noncompliance.

The first obligation represents Policy A. In this case, the "target" is the NGSI notification received from the Context Broker. The action that the DC is required to perform in this policy is "aggregate" (combine data individual values into one). In addition, we define constraints applied to said action: the use of the terms "leftOperand", "operator", and "rightOperand" allow us to define the logical

constraints to be applied. The values presented in this fragment of code means that the DC is obliged to aggregate the notifications received at least every 15 min before generating an output. Finally, as a consequence, we establish that a kill signal for stopping the running program associated with this rule will be sent ("killJob" action). As can be seen, a similar approach is followed in the second obligation, which represents Policy B. The aim of using ODRL is to provide dynamic capabilities so as to enable the PTP to generate an extended automaton on the basis of the policies that will run on the uPDP.

Listing 1: ODRL Specification of Policy A and Policy B.

```
{
  "@context": ["http://www.w3.org/ns/odrl.jsonld",
    "http://keyrock.fiware.org/FIDusageML/profile/FIDusageML.jsonld"],
  "@type": "Set",
  "uid": " http://keyrock.fiware.org/FIDusageML/policy:1010",
  "profile": "http://keyrock.fiware.org/FIDusageML/profile",
  "obligation": [{
       "target": "http://orion.fiware.org/NGSInotification",
       "action": "aggregate",
       "constraint": [{
         "leftOperand":"WindowNotification",
         "operator": "gt",
         "rightOperand": {
              "@value": "PT15M",
              "@type": "xsd:duration"}
       }],
       "consequence": [{
         "action": "killJob",
         "value": "http://orion.fiware.org/NGSIkilljob"
        }],
    },
    {
       "target": "http://orion.fiware.org/NGSInotification",
       "action": "NGSIEventLimit",
       "constraint": [{
         "leftOperand": "NGSIevent",
         "operator": "lt",
         "rightOperand": {
              "@value": "200",
              "@type": "xsd:integer" }},
       {
         "leftOperand":"WindowNotification",
         "operator": "gt",
         "rightOperand": {
              "@value": "PT1M",
              "@type": "xsd:duration"}
       }],
       "consequence": [{
         "action": "unsubscribe",
         "value": "http://orion.fiware.org/NGSIunsubscribe"
       }],
    },
  ]
}
```

The code that the PTP generates from the ODRL specification is showed in Listing 2. This program is in charge of correlating complex events generated by the DC's operations with the rules regarding data usage. The value `countPattern` represents the maximum number of events (`timesOrMore`) that can be received in the time window (`within`) specified in Policy A. Whenever the program detects a behavior that fails to comply with this rule, it immediately passes the control to the PXP (`Signals.createAlert`) in order to send the control signal to the corresponding component according to the punishment previously defined. A similar pattern is applied to Policy B.

Listing 2: uPDP code generated from the ODRL Specification by the PTP.

```
val operationStream : DataStream[ExecutionGraph] = stream
  .filter(_.isRight)
  .map(_.right.get)
  .flatMap(_.msg.split(" -> "))
  .map(ExecutionGraph)

// Entity Stream
val entityStream : DataStream[Entity] = stream
  .filter(_.isLeft)
  .map(_.left.get )
  .flatMap(_.entities)

// First pattern: At least N events in T. Any other time
val countPattern = Pattern.begin[Entity]("events" )
  .timesOrMore(Policies.numMaxEvents+1)
  .within(Time.seconds(Policies.facturationTime))
CEP.pattern(entityStream, countPattern)
  .select(events =>
  Signals.createAlert(Policy.COUNT_POLICY, events, Punishment.UNSUBSCRIBE))

// Second pattern: Source -> Sink. Aggregation TimeWindow
val aggregatePattern = Pattern.begin[ExecutionGraph]
("start", AfterMatchSkipStrategy.skipPastLastEvent())
  .where(Policies.executionGraphChecker(_, "source"))
  .notFollowedBy("middle")
  .where(Policies.executionGraphChecker(_,"aggregation",Policies.aggregateTime))
  .followedBy("end")
  .where(Policies.executionGraphChecker(_, "sink"))
  .timesOrMore(1)
CEP.pattern(operationStream, aggregatePattern).select(events =>
  Signals.createAlert(Policy.AGGREGATION_POLICY, events, Punishment.KILL_JOB))
```

5.3. Data Processing and Policy Enforcement

Once the policies have been defined, the DC may start to deploy processing jobs on their own infrastructure with the aim of extracting value from the supermarket data received. In order to validate the two policies defined by the DP, we have created two sample jobs that operate on the DP's data:

Job I: Direct sinking of ticket data

The first job reads the data received from the DP and sends them somewhere else, outside of the scope of the data usage architecture, in which operations on data are not monitored. Since this use allows the DC to process each piece of data individually, without prior aggregation, it is a clear violation of Policy A. When the DC deploys this job in the Flink cluster, an Execution Graph is calculated from the program code. The Execution Graph is the central data structure that coordinates the distributed execution of a data flow. It contains a representation of each parallel task, each intermediate stream, and the communication among them. Figure 6 shows the Execution Graph generated for Job I, in which all the operations performed on data are included. The first item in the Execution Graph is a Custom Source. It indicates that the program uses a custom connector as an input for receiving

data streams, in this case, the custom source is the one provided by the FIWARE Cosmos GE. Since no additional operations are performed on data, the Source is immediately followed by a Data Sink, which consumes Data Streams and forwards them to files, sockets, external systems, etc. or prints them on the standard output.

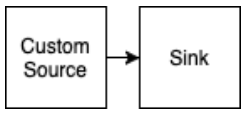

Figure 6. Simplified Execution Graph of Job I.

The log containing the chain of operations (as shown in Listing 3) generated by the DC's Flink processing engine is sent to the uPDP, which will detect that the Execution Graph contains no aggregation of data, thus failing to comply with Policy A. The uPDP will inform the PXP of this violation, which will send the corresponding control signal described in the policy; in this case, terminating the job as a punishment for noncompliance.

One major concern about using the Execution Graph for policy enforcement is ensuring that it has been correctly generated. The processing engine itself is in charge of detecting all the operations performed within a program and generating the Execution Graph. The integrity of the processing engine relies on the use of trusted environments, achieved through IDS connectors, in which no alteration of the run-time environment is allowed. Thus, all the operations that the processing engine detects it must perform within a certain program are reflected on the Execution Graph and no operation is overlooked or disregarded. The Execution Graph is reflected on a log which is then sent to the uPDP. As far as the integrity, confidentiality and authenticity of the in-transit logs is concerned, it is ensured by means of the mechanisms explained at the end of Section 4.

Listing 3: Sample Execution Graph Log generated by the DC's Flink processing engine for Job I.

```
org.apache.flink.runtime.executiongraph.ExecutionGraph - Source: Custom Source -> Sink: Print to Std. Out
    (1/1) (5dd7fd1626577f325e61fe1effc996c2) switched from SCHEDULED to~DEPLOYING.
```

Job II: Calculating average ticket price

The second job calculates the average ticket price for all the purchases of each store every hour. This operation is an aggregation of data so, when the Execution Graph is checked by the uPDP, it will be verified that it complies with Policy A. Figure 7 represents the Execution Graph generated for this use case. The logs that are sent to the uPDP containing this Execution Graph are shown in Listing 4.

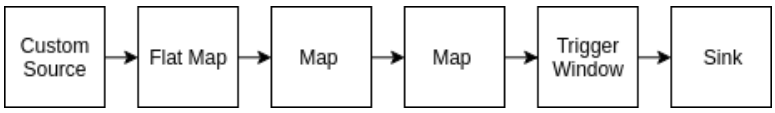

Figure 7. Simplified Execution Graph of Job II.

Listing 4: Execution Graph Log generated by the DC's Flink processing engine for Job II.

```
2019-07-18 11:07:59.773 [flink-akka.actor.default-dispatcher-2] INFO
    org.apache.flink.runtime.executiongraph.ExecutionGraph - Source: Custom Source -> Flat Map -> Map ->
    Map (1/1) (b613fa2767c444587f05a3502b8fc7b0) switched from SCHEDULED to~DEPLOYING.

2019-07-18 11:07:59.773 [flink-akka.actor.default-dispatcher-2] INFO
    org.apache.flink.runtime.executiongraph.ExecutionGraph - Deploying Source: Custom Source ->
    Flat Map -> Map -> Map (1/1) (attempt #0) to~fc98c42cd3a0

2019-07-18 11:07:59.778 [flink-akka.actor.default-dispatcher-2] INFO
  org.apache.flink.runtime.executiongraph.ExecutionGraph - TriggerWindow(TumblingProcessingTimeWindows
  (15000), AggregatingStateDescriptor{name=window-contents, defaultValue=null,
  serializer=org.fiware.cosmos.orion.flink.cep.examples.example1.AveragePrice$$anon$26$$anon$11@3bbaf7ca},
  ProcessingTimeTrigger(), AllWindowedStream.aggregate(AllWindowedStream.java:475)) ->
  Sink: Print to Std. Out (1/1) (de683f055949331f5602e94b306ae10d) switched from SCHEDULED to~DEPLOYING.
```

Besides sending the Execution Graph logs to the uPDP, each time the DC receives information from one ticket, this event is logged and sent to the uPDP as well (as shown in Listing 5). If the uPDP detects that the DC has received more tickets than the amount specified by Policy B (200), the PXP will be notified and will enforce the corresponding punishment (i.e., removing the subscription to the tickets' data).

Listing 5: Sample notification Log generated by the DC's Flink processing engine for Job II.

```
2019-07-18 15:15:32.032 [nioEventLoopGroup-3-2] INFO
 org.fiware.cosmos.orion.flink.connector.OrionHttpHandler -
  {"creationTime":1563462932031,"fiwareService":"555","fiwareServicePath": "application/json;
  charset=utf-8","entities": [{"id":"ticket","type":"ticket","attrs":{"_id":{"type":"String","value":75,
  "metadata":{}},"items":{"type":"object","value":[{"net_am":4.99,"n_unit":1,"desc":"BREAD"},{"net_am":
  5.5,"n_unit":2,"desc":"PIZZA HAM/CHEESE"},{"net_am":2.39,"n_unit":1,"desc":"FRANKFURT SAUSAGES"},
  {"net_am":0.05,"n_unit":1,"desc":"SHOPPING BAG"}],"metadata":{}},"mall":{"type":"String","value":1,
  "metadata":{}},"date":{"type":"date","value":"01/14/2016","metadata":{}},"client":{"type":"int","value":
  77053280208,"metadata":{}}}}],"subscriptionId":"5d308d139d5b4d3e64685da0"}
```

Overall, the case study presented in this section, including the deployment of both DC's jobs, shows that the data usage architecture provides a way of verifying that the DC complies with a set of predefined policies by the DP and of executing punishments in case of noncompliance.

5.4. Results

This subsection presents a series of metrics carried out in the case study presented that aim to calculate the enforcement time of the policies defined. To this end, the two jobs presented were deployed, achieving noncompliance conditions for both of them.

In accordance with the scheme presented in the implementation section, we define the deployment of all the components as follows: every building block of the DC and DP was deployed using Docker containers (Docker: https://www.docker.com). However, in order to test the Usage Control policies and obtain more accurate metrics, we deployed the DC and the DP in different platforms. On the one hand, the DP's containers were placed inside of a VM (Virtual Machine) in an Edge Computing Infrastructure using OpenStack (OpenStack: https://www.openstack.org), this VM has the following features: 2VCPU's, 4 GB RAM, 40 GB Disk. On the other hand, the DC's containers were placed in a local server located in the same network as the DP's VM with the following specifications: 4.0 GHz Intel core I7 CPU with 8 GB RAM and 256 SSD Disk. This deployment allowed testing different policies and measure overhead time. The workflow of submitting a job on the DC's side, detecting the policy noncompliance, and enforcing the due punishment was repeated N times (N = 100) for each policy. Through the system logs generated by the DC's and DP's containers, three different metrics were calculated:

- **Decision time** (T_d): Time between the policy infringement on the DC's side and the detection of said infringement at the uPDP on the DP's side.
- **Execution time** (T_x): Time between the policy infringement detection at the uPDP and the execution of the punishment at the PXP.
- **Total enforcement time** (T_t): Sum of T_d and T_x.

Table 2 summarizes the results obtained for each policy and interval, including the mean time (M) and the standard deviation (SD). The results for each iteration can be seen in Figures 8 and 9.

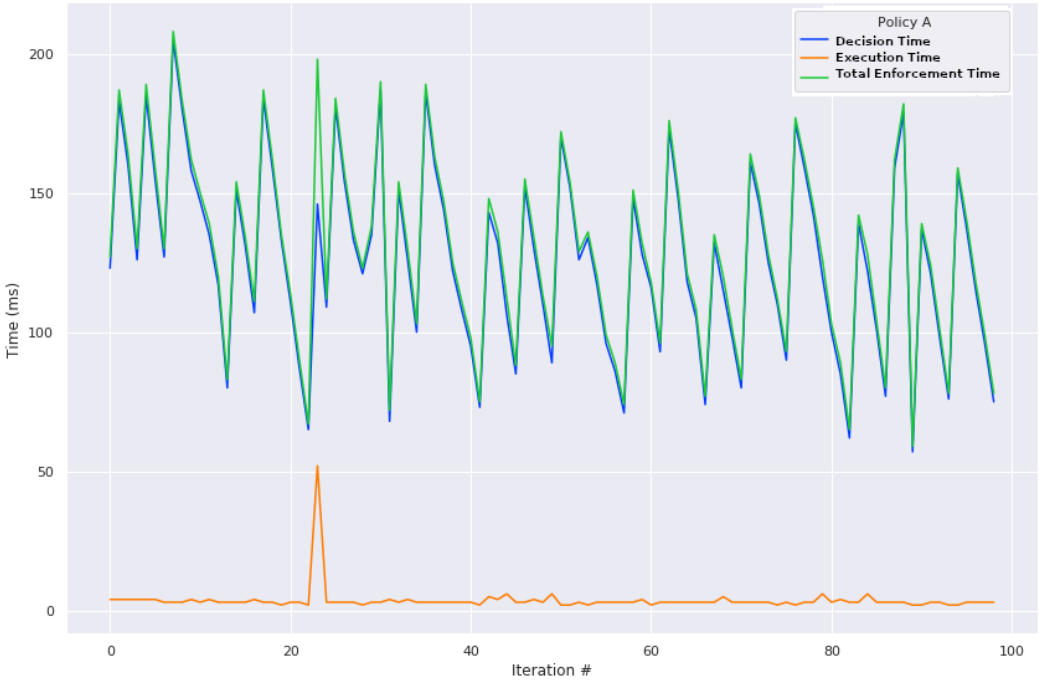

Figure 8. Policy A enforcement metrics.

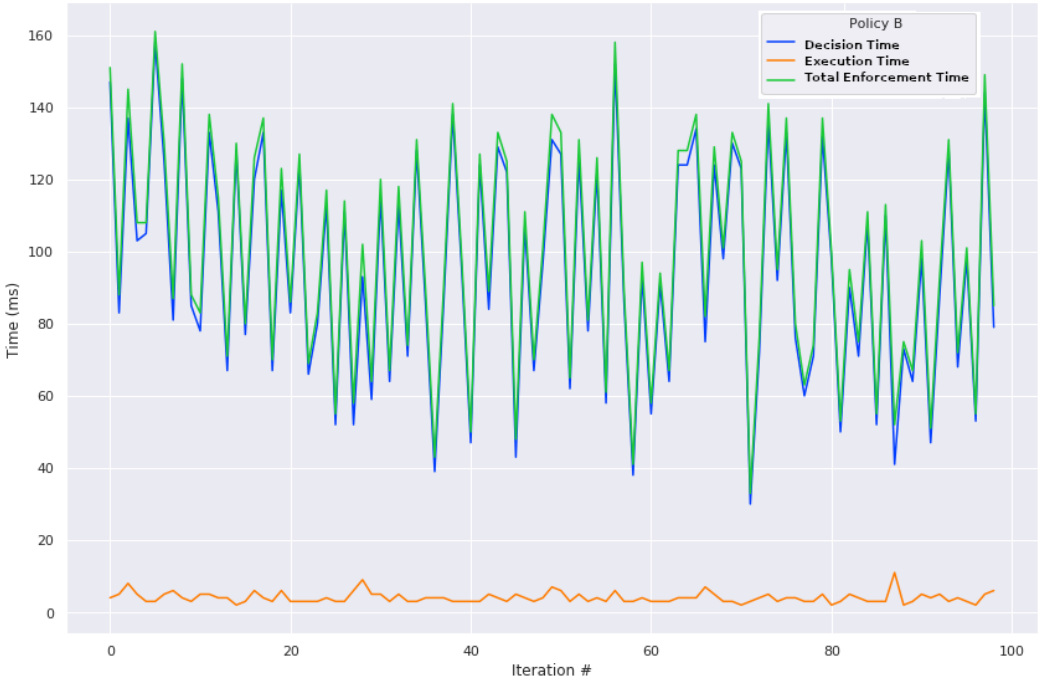

Figure 9. Policy B enforcement metrics.

Table 2. Summary of enforcement time metrics.

	Policy A		**Policy B**	
	M	**SD**	**M**	**SD**
T_d (ms)	125.8	33.8	94.6	31.4
T_x (ms)	3.7	5.0	4.0	1.5
T_t (ms)	129.5	34.5	98.6	31.6

As is apparent from the results shown in Table 2, the times registered for T_d are significantly larger than those recorded for T_x. The main reason for this difference is the fact that T_d involves generating the logs on the DC's side, writing them on the processing engine's log file and sending them to the uPDP, which will receive them after they are first verified by the PEP. By contrast, the T_x is very low since the PXP is embedded in the same program that the uPDP is running, which means that the delay introduced stems from the time it takes to receive an acknowledgement from the control signal sent to the system in charge of performing the actual punishment (to the Context Broker in order to remove a subscription, or to the DC's processing engine to cancel the job).

Furthermore, the slight difference in measurements for T_x between policies A and B draws from the difference in response times between the Context Broker and the processing engine on the DC's side. On the other hand, the difference in T_d between policies A and B is due to the fact that in the former, the noncompliance is detected by inspecting the Execution Graph, which is received by the uPDP as a single log, whereas in the latter, the uPDP needs to inspect the message history to confirm that the notification limit established within the policy has been exceeded, which is a more costly operation.

As far as the total time (T_t) is concerned, the T_x can be neglected for its calculation if the punishment does not involve interacting with stakeholders outside a shared network (the Context Broker, for instance). Otherwise, the network latency needs to be taken into account. This holds true for the T_d as well, since typically the DC and the DP are in physically separate networks. The results obtained for T_t fall within a reasonable range of values for most use cases, in which new data are generated every second or few seconds. However, in scenarios in which new data are published within milliseconds, there could be a period between the infringement of the rules and the enforcement of the punishments in which new data are unduly received by the DC. In order to verify that this was not the case, we tested our solution under different stress situations. We deployed the use case scenario using different frequencies of generation of new data. The main goal was to corroborate that no additional data events arrived from the moment an infringement was committed and the moment that the due punishment was executed. The use case scenario was tested for data generation periods ranging from 5 s to 25 ms (5 s, 1 s, 500 ms, 250 ms, 100 ms, 50 ms, 25 ms), repeating each simulation 100 times. We found that all the situations of data infringement were detected, and the appropriate punishment was enforced in due time in 100% of the cases. Thus, no new information was unduly received after failing to comply with a certain policy. Nevertheless, it is worth pointing out that for periods under 25 ms, we have to consider the throughput limit of the SDS (in our case the Context Broker) since this component is the one that determines the actual speed at which data will be sent to the endpoints that are subscribed to new data.

Prior works [33,34] have also collected a series of metrics to validate their models. For example, Marra et al. [34] determine if the performance of their Usage Control system is higher whether it is applied to local or remote attributes. Furthermore, instead of measuring enforcement time, Wu et al. [33] focus on performance at the Access Control level. Although an exact comparison between said models and the one presented in this work cannot be performed, since none of the works found in the existing literature provide measurements of the decision and enforcement times, it can be seen that the measurements for delay and response times are in the same order of magnitude.

6. Conclusions and Future Work

The implementation of the architecture presented in this paper provides a comprehensive and affordable solution for providing access and Usage Control in industrial data ecosystems. One of the advantages of this proposal stems in the fact that it is suitable for being implemented in a wide range of different scenarios since it is a technology-agnostic solution. This characteristic, along with its fine-grained Access and Usage Control capabilities and its multi-actor architecture contributes to fill the gap in the existing literature.

Moreover, this piece of research also presents an implementation of the referenced architecture using FIWARE Generic Enablers that completes the previously proposed implementation of IDS architecture [17]. The implementation presented has been validated with a use case in the food industry, presenting a series of metrics of the response time of policy compliance verification and punishment enforcement. The data Usage Control components developed in the scope of this work (uPDP, PXP and PTP) have been proposed and accepted as a new Generic Enabler in the FIWARE catalogue.

As a conclusion to this work, we identify the need for defining a policy specification language capable of handling the fine-grained policy definitions to provide data Usage Control capabilities in Industry 4.0. Furthermore, we consider that this architecture could be extended to become GDPR-compliant by introducing the GDPR regulation rules inside the definition of the policies and deploying them inside this architecture. The work presented in [59] provides a preliminary version of the definition of ODRL policy specification oriented to GDPR. This is still an ongoing research topic. We consider that providing an ODRL vocabulary for GDPR and some policy examples can lead to the inclusion of this vocabulary as an ODRL extension in the W3C working groups and further be supported by the community interested in this topic. We will focus our future efforts on the Policy Specification Language definition, conceived as an extension of ODRL, as well as the definition of a common vocabulary that allows standardization and identification of the events and traces of the system for the different processing engines. Additionally, we intend to develop new and more complex tests that allow us to extract additional metrics in the scope of data protection.

Another possible area of future research would be to investigate the integration of Blockchain technologies within the Usage Control proposed architecture. Data Providers could store the Usage Control policies that are applied to specific data and to specific Data Consumers in the distributed ledger and check them whenever the latter perform an operation. Blockchain scales best with lightweight information types, so the data itself should remain out of the ledger and only the metadata representing the data to which Usage Control policies are applied should be stored in the ledger. This approach would enhance trust and transparency over data accountability and traceability between consumers and providers. However, further research needs to be performed on how to integrate Blockchain in dynamic environments with high density of requests such as those in IoT scenarios.

Author Contributions: Conceptualization, A.M.-A., S.L.-P., Á.A., A.P., J.S. and G.H.; Formal analysis, A.M.-A., S.L.-P., J.S. and G.H.; Investigation, A.M.-A., S.L.-P., and A.P.; Methodology, A.M.-A., S.L.-P., Á.A. and A.P.; Software, A.M.-A., S.L.-P., and A.P.; Supervision, A.M.-A., Á.A., J.S. and G.H.; Validation, A.M.-A., S.L.-P., Á.A. and A.P.; Visualization, A.M.-A. and S.L.-P.; Writing—original draft, A.M.-A., S.L.-P., Á.A. and A.P.; Writing—review & editing, A.M.-A., S.L.-P., Á.A. and A.P.; Funding acquisition, A.M.-A., Á.A., J.S. and G.H.; Project administration, A.M.-A., S.L.-P., Á.A. and J.S.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Jeschke, S.; Brecher, C.; Meisen, T.; Özdemir, D.; Eschert, T. Industrial Internet of Things and Cyber manufacturing systems. In *Ind. Internet Things*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 3–19.
2. Lu, Y. Industry 4.0: A survey on technologies, applications and open research issues. *J. Ind. Inf. Integr.* **2017**, *6*, 1–10. [CrossRef]

3. Mosavi, A.; Vaezipour, A. *Developing Effective Tools for Predictive Analytics and Informed Decisions*; Technical Report; University of Tallinn: Tallinn, Estonia, 2013.
4. Tiwari, V. Study of Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions. *Int. J. Adv. Res. Comp. Sci.* **2016**, *7*, 65–84.
5. Kagermann, H.; Helbig, J.; Hellinger, A.; Wahlster, W. *Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0: Securing the Future of German Manufacturing Industry; Final Report of the Industrie 4.0 Working Group*; Forschungsunion: Essen, Germany, 2013 .
6. Mosavi, A.; Lopez, A.; Varkonyi-Koczy, A.R. Industrial applications of big data: State of the art survey. In *International Conference on Global Research and Education*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 225–232.
7. Sandhu, R.S.; Samarati, P. Access control: Principle and practice. *IEEE Comm. Mag.* **1994**, *32*, 40–48. [CrossRef]
8. Sandhu, R.; Park, J. Usage Control: A Vision for Next Generation Access Control. In *Computer Network Security, Proceedings of the 2nd International Workshop on Mathematical Methods, Models, and Architectures for Computer Network Security, MMM-ACNS 2003, St. Petersburg, Russia, 21–23 September 2003*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 17–31.
9. Voigt, P.; von dem Bussche, A. *The EU General Data Protection Regulation (GDPR). A Practical Guide*; Springer: Berlin/Heidelberg, Germany, 2017. [CrossRef]
10. Otto, B.; Lohmann, S.; Steinbuss, S.; Teuscher, A. *IDS Reference Architecture Model Version 2.0*; Technical Report; Fraunhofer: Munich, Germany, 2018 .
11. Bettini, C.; Jajodia, S.; Wang, X.S.; Wijesekera, D. Provisions and Obligations in Policy Rule Management. *J. Netw. Syst. Manag.* **2003**, *11*, 351–372. [CrossRef]
12. OASIS Standard. eXtensible Access Control Markup Language (XACML) Version 3.0. Available online: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf (accessed on 3 June 2019).
13. Alonso, Á.; Fernández, F.; Marco, L.; Salvachúa, J. IAACaaS: IoT Application-Scoped Access Control as a Service. *Futur. Internet* **2017**, *9*, 64. [CrossRef]
14. Fernández, F.; Alonso, Á.; Marco, L.; Salvachúa, J. A model to enable application-scoped access control as a service for IoT using OAuth 2.0. In Proceedings of the 2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN), Paris, France, 7–9 March 2017; pp. 322–324. [CrossRef]
15. Munoz-Arcentales, A.; López-Pernas, S.; Pozo, A.; Álvaro, A.; Salvachúa, J.; Huecas, G. An Architecture for Providing Data Usage and Access Control in Data Sharing Ecosystems. *Procedia Comput. Sci.* **2019**, *160*, 590–597. [CrossRef]
16. Ravidas, S.; Lekidis, A.; Paci, F.; Zannone, N. Access control in Internet-of-Things: A survey. *J. Netw. Comp. Appl.* **2019**, *144*, 79–101. [CrossRef]
17. Alonso, Á.; Pozo, A.; Cantera, J.M.; la Vega, F.; Hierro, J.J. Industrial Data Space Architecture Implementation Using FIWARE. *Sensors* **2018**, *18*, 2226. [CrossRef]
18. Xu, L.D.; Duan, L. Big data for cyber physical systems in industry 4.0: A survey. *Ent. Inf. Syst.* **2019**, *13*, 148–169. [CrossRef]
19. Lee, J.; Kao, H.A.; Yang, S. Service innovation and smart analytics for industry 4.0 and big data environment. *Procedia Cirp* **2014**, *16*, 3–8. [CrossRef]
20. Yin, S.; Kaynak, O. Big data for modern industry: Challenges and trends [point of view]. *Proc. IEEE* **2015**, *103*, 143–146. [CrossRef]
21. Mourtzis, D.; Vlachou, E.; Milas, N. Industrial Big Data as a result of IoT adoption in manufacturing. *Procedia Cirp* **2016**, *55*, 290–295. [CrossRef]
22. Gölzer, P.; Cato, P.; Amberg, M. Data Processing Requirements of Industry 4.0-Use Cases for Big Data Applications. In Proceedings of the ECIS 2015, Münster, Germany, 26–29 May 2015.
23. Gokalp, M.O.; Kayabay, K.; Akyol, M.A.; Eren, P.E.; Koçyiğit, A. Big data for industry 4.0: A conceptual framework. In Proceedings of the 2016 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 15–17 December 2016; pp. 431–434.
24. Osman, A.M.S. A novel big data analytics framework for smart cities. *Future Gener. Comp. Syst.* **2019**, *91*, 620–633. [CrossRef]
25. Zhu, J.Y.; Tang, B.; Li, V.O. A five-layer architecture for big data processing and analytics. *Int. J. Big Data Int.* **2019**, *6*, 38–49.

26. NGSI-LD, E. Context Information Management (CIM) and Application Programming Interface (API). *ETSI GS CIM* **2018**, *4*, V1.
27. Russello, G.; Dulay, N. xDUCON: Cross Domain Usage Control through Shared Data Spaces. In Proceedings of the 2009 IEEE International Symposium on Policies for Distributed Systems and Networks, London, UK, 20–22 July 2009; pp. 178–181. [CrossRef]
28. Russello, G.; Dulay, N. xDUCON: Coordinating Usage Control Policies in Distributed Domains. In Proceedings of the 2009 Third International Conference on Network and System Security, Gold Coast, QLD, Australia, 19–21 October 2009; pp. 246–253. [CrossRef]
29. Cerbo, F.D.; Some, D.; Gomez, L.; Trabelsi, S. PPL v2.0: Uniform Data Access and Usage Control on Cloud and Mobile. In Proceedings of the 2015 IEEE/ACM 1st International Workshop on TEchnical and LEgal aspects of data pRivacy and SEcurity, Florence, Italy, 18 May 2015; pp. 2–7. [CrossRef]
30. Ardagna, C.A.; Bussard, L.; De Capitani di Vimercati, S.; Neven, G.; Pedrini, E.; Paraboschi, S.; Preiss, F.; Samarati, P.; Trabelsi, S.; Verdicchio, M. PrimeLife Policy Language. In Proceedings of the W3C Work Access Control Appl. Scenar., Luxembourg, 17–18 November 2009.
31. Jiao, D.; Lianzhong, L.; Ting, L.; Shilong, M. Realization of UCON Model Based on Extended-XACML. In Proceedings of the 2011 International Conference on Future Computer Sciences and Application, Hong Kong, China, 18–19 June 2011; pp. 90–93. [CrossRef]
32. Lazouski, A.; Mancini, G.; Martinelli, F.; Mori, P. Usage control in cloud systems. In Proceedings of the 2012 International Conference for Internet Technology and Secured Transactions, London, UK, 10–12 December 2012; pp. 202–207.
33. Wu, J.; Dong, M.; Ota, K.; Tariq, M.; Guo, L. Cross-Domain Fine-Grained Data Usage Control Service for Industrial Wireless Sensor Networks. *IEEE Access* **2015**, *3*, 2939–2949. [CrossRef]
34. Marra, A.L.; Martinelli, F.; Mori, P.; Saracino, A. Implementing Usage Control in Internet of Things: A Smart Home Use Case. In Proceedings of the 2017 IEEE Trustcom/BigDataSE/ICESS, Sydney, NSW, Australia, 1–4 August 2017; pp. 1056–1063. [CrossRef]
35. Bertolino, A.; Calabrò, A.; Lonetti, F.; Sabetta, A. Glimpse: A generic and flexible monitoring infrastructure. In Proceedings of the 13th European Workshop on Dependable Computing (EWDC), Pisa, Italy, 11–12 May 2011.
36. Barsocchi, P.; Calabrò, A.; Ferro, E.; Gennaro, C.; Marchetti, E.; Vairo, C. Boosting a low-cost smart home environment with usage and access control rules. *Sensors* **2018**, *18*, 1886. [CrossRef] [PubMed]
37. Gkioulos, V.; Rizos, A.; Michailidou, C.; Mori, P.; Saracino, A. Enhancing Usage Control for Performance: An Architecture for Systems of Systems. In *Comp. Sec.*; Katsikas, S.K., Cuppens, F., Cuppens, N., Lambrinoudakis, C., Antón, A., Gritzalis, S., Mylopoulos, J., Kalloniatis, C., Eds.; Springer: Berlin/Heidelberg, Germany, 2019; pp. 69–84.
38. Martinelli, F.; Michailidou, C.; Mori, P.; Saracino, A. Managing QoS in Smart Buildings Through Software Defined Network and Usage Control. In Proceedings of the 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Kyoto, Japan, 11–15 March 2019; pp. 626–632.
39. Petković, M.; Prandi, D.; Zannone, N. Purpose control: Did you process the data for the intended purpose? In *Workshop on Secure Data Management*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 145–168.
40. Poullet, Y. EU data protection policy. The Directive 95/46/EC: Ten years after. *Comput. Law Secur. Rev.* **2006**, *22*, 206–217. [CrossRef]
41. Bartolini, C.; Daoudagh, S.; Lenzini, G.; Marchetti, E. Towards a lawful authorized access: A preliminary GDPR-based authorized access. In Proceedings of the ICSOFT 2019, Prague, Czech Republic, 26–28 July 2019; pp. 26–28.
42. Bartolini, C.; Daoudagh, S.; Lenzini, G.; Marchetti, E. GDPR-Based User Stories in the Access Control Perspective. In *Quality of Information and Communications Technology, Proceedings of the 12th International Conference, QUATIC 2019, Ciudad Real, Spain, 11–13 September 2019*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 3–17.
43. Calabró, A.; Daoudagh, S.; Marchetti, E. Integrating Access Control and Business Process for GDPR Compliance: A Preliminary Study. In Proceedings of the ITASEC 2019, Pisa, Italy, 13–15 February 2019.
44. Arfelt, E.; Basin, D.; Debois, S. Monitoring the GDPR. In *Comp. Sec.–ESORICS 2019*; Sako, K., Schneider, S., Ryan, P.Y.A., Eds.; Springer: Berlin/Heidelberg, Germany, 2019; pp. 681–699.

45. Basin, D.; Harvan, M.; Klaedtke, F.; Zălinescu, E. MONPOLY: Monitoring Usage-Control Policies. In *Runt. Verif.*; Khurshid, S., Sen, K., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 360–364.
46. Neisse, R.; Steri, G.; Nai-Fovino, I. A Blockchain-Based Approach for Data Accountability and Provenance Tracking. In *Proceedings of the 12th International Conference on Availability, Reliability and Security, ARES '17*; Association for Computing Machinery: New York, NY, USA, 2017. [CrossRef]
47. Outchakoucht, A.; Hamza, E.; Leroy, J.P. Dynamic access control policy based on blockchain and machine learning for the internet of things. *Int. J. Adv. Comput. Sci. Appl.* **2017**, *8*, 417–424. [CrossRef]
48. Ouaddah, A.; Abou Elkalam, A.; Ait Ouahman, A. FairAccess: A new Blockchain-based access control framework for the Internet of Things. *Sec. Comm. Netw.* **2016**, *9*, 5943–5964. [CrossRef]
49. Panian, Z. Some practical experiences in data governance. *World Acad. Sci. Eng. Technol.* **2010**, *62*, 939–946.
50. Paci, F.; Squicciarini, A.; Zannone, N. Survey on access control for community-centered collaborative systems. *ACM Comp. Surv.* **2018**, *51*, 1–38. [CrossRef]
51. Mahmudlu, R.; den Hartog, J.; Zannone, N. Data governance and transparency for collaborative systems. In *Data and Applications Security and Privacy XXX, Proceedings of the 30th Annual IFIP WG 11.3 Conference, DBSec 2016, Trento, Italy, 18–20 July 2016*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 199–216.
52. European Data Protection Supervisor. European Data Protection Supervisor Glossary. 2019. Available online: https://edps.europa.eu/data-protection/data-protection/glossary/d_en (accessed on 3 June 2019).
53. McRoberts, M.; Rodriguez Doncel, V. *Open Digital Rights Language (ODRL) Ontology*; Technical Report; W3C: Cambridge, MA, USA, 2014 .
54. Open Mobile Alliance . NGSI Context Management. Available online: http://www.openmobilealliance\.org/release/NGSI/V1_0-20120529-A/OMA-TS-NGSI_Context_Management-V1_0-20120529-A.pdf (accessed on 8 July 2019).
55. Digital CEF. Context Broker, Make Data-Driven Decisions in Real Time, at the Right Time. Available online: https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/Context+Broker (accessed on 3 September 2019).
56. Rescorla, E. HTTP Over TLS; RFC 2818, RFC Editor; California, United States, 2000. Available online: https://tools.ietf.org/html/rfc2818 (accessed on 1 May 2020).
57. Teixeira, A.; Pérez, D.; Sandberg, H.; Johansson, K.H. Attack models and scenarios for networked control systems. In Proceedings of the 1st International Conference on High Confidence Networked Systems, Beijing, China, 17–18 April 2012; pp. 55–64.
58. Steyskal, S.; Polleres, A. Towards Formal Semantics for ODRL Policies. In *Rule Tech. Found., Tools, App.*; Bassiliades, N., Gottlob, G., Sadri, F., Paschke, A., Roman, D., Eds.; Springer: Berlin/Heidelberg, Germany, 2015 ; pp. 360–375.
59. De Vos, M.; Kirrane, S.; Padget, J.; Satoh, K. ODRL policy modelling and compliance checking. In *Rules and Reasoning, Proceedings of the Third International Joint Conference, RuleML+RR 2019, Bolzano, Italy, 16–19 September 2019*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 36–51.