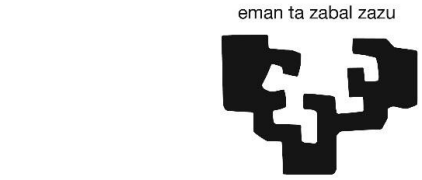

ADVANCES IN SOVEREIGN DATA SHARING: IDENTIFICATION AND ASSESSMENT OF THE MAIN FEATURES OF DISTRIBUTED USAGE CONTROL SOLUTIONS AND IMPROVEMENTS IN THE POLICY QUALITY

Ph.D. Thesis in Mobile Network Information and Communication Technologies

Gonzalo Gil Inchaurza

Supervisors:

Dra. Mariví Higuero Aperribay

Dr. Aitor Arnaiz Irigaray

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

Ph.D. Thesis

ADVANCES IN SOVEREIGN DATA SHARING: IDENTIFICATION AND ASSESSMENT OF THE MAIN FEATURES OF DISTRIBUTED USAGE CONTROL SOLUTIONS AND IMPROVEMENTS IN THE POLICY QUALITY

Presented by

Gonzalo Gil Inchaurza

Bilbao, 2023

Tekniker
MEMBER OF BASQUE RESEARCH & TECHNOLOGY ALLIANCE

"Without commitment,
you will never start,
but without consistency,
you will never finish."

Denzel Washington

Agradecimientos

Echando la mirada atrás me doy cuenta de que tengo que estar muy agradecido por la gente que ha estado a mi lado de un modo u otro durante este largo camino que ha supuesto el realizar esta tesis.

Querría empezar agradeciendo a Tekniker, por darme los medios y la confianza para poder llevar a cabo esta tesis. En especial, a todos mis compañeros de SII. A Aitor, por su disposición a realizar el esfuerzo necesario para que la tesis fuera hacia delante pese a las dificultades que aparecían. A Francis, por todo lo que he aprendido desde que entré en Tekniker. Y a Roberto, Nacho, Ricardo, Izaskun, Carlos, Maite, etc. por el apoyo. No me quiero olvidar de Marivi, gracias por todo el esfuerzo dedicado y el aprendizaje adquirido en el ámbito de la investigación.

En el ámbito más personal, agradecer a toda mi familia. A mis abuelos, por hacer que nunca me falte de nada. A mis tíos y primos, por preocuparos por mí en cada momento. A mi padre, por todo el cariño y la confianza que me has dado en cada uno de los pasos que he ido dando a lo largo de mi vida. A mi madre, ya que, si no fuera por todo tu esfuerzo, yo no estaría aquí en este momento. Y a mi hermana, porque con nuestros más y nuestros menos siempre has estado cuando lo he necesitado.

Gracias a ti Ángela, por toda la paciencia que has tenido, por entenderme en cada momento, por apoyarme cuando lo necesitaba y, sobre todo, por hacerme disfrutar más de la vida. También, a toda su familia, por hacerme sentir uno más.

Finalmente, agradecer a mis amigos de toda la vida, Pablo, Javi, Luis, Ander y Adri, por estar ahí siempre. A Patri, Julen, Sergio, Iñigo y

Javi por hacer este camino más ameno con cada viaje, cada café y cada comida. A Nere y Unai, por cada uno de los planes. A Ainhoa por el poteo y picoteo. Y a Iñi, Adri, Clau, Marina, Amaia, Asier, Juan y Andoni por los entrenamientos de cada semana.

Abstract

The amount of data being generated around the world is growing year by year. At the same time, data-driven technologies are enabling organizations to improve their activities. These technologies, require more and more data to remain competitive, especially from other organizations. This has made data and its sharing among organizations a key resource for economic growth, competitiveness and innovation. Nevertheless, a critical issue that is preventing data sharing worldwide is the reluctance that organizations have to share their data if their self-determination about the usage of their data is not granted. This is referred to as the data sovereignty concern.

At the present time, current initiatives among which the International Data Spaces Association (IDSA) stand out, work in the definition and design a distributed, open, interoperable, and sovereign infrastructure of services in which different organizations can collaborate and benefit from data sharing. To grant data sovereignty, the IDSA proposes Distributed Usage Control (DUC) a particularization of Usage Control (UC) for data sharing scenarios that extends Access Control (AC) to control what must happen to data through its life cycle.

This thesis seeks to strengthen data sharing through advances in data sovereignty providing new tools to increase the widespread adoption of DUC solutions. In particular, this thesis provides a novel approach on the identification and assessment of the features that DUC solutions must support to ensure data sovereignty. In turn, this thesis provides improvements in the quality of the policies defined to restrict the usage of the data in DUC solutions.

First, different DUC solutions already exist. However, as they have different goals, they present different, specific features. This makes it difficult to identify the features that DUC solutions must support to ensure data sovereignty. In turn, there is no framework for the assessment to what extent existing DUC solutions ensure data sovereignty. This thesis presents a framework to identify and assess the features that DUC solutions must support to meet data sovereignty requirements in the context of data sharing. This framework should take a big step towards the adoption of DUC solutions. It sets the path to ensure data sovereignty through DUC. Furthermore, it enables the identification of strengths and weaknesses of DUC solutions regarding data sovereignty. This framework has been validated applying it to the most widespread DUC solutions. As a result, their capabilities and limitations have been identified. This will enable service provides to select and deploy the most suitable solution for a particular scenario. Furthermore, it will enable software providers to identify existing weaknesses from these DUC solutions to improve them.

Second, one common limitation identified from all existing DUC solutions is related to the quality of the policies that define restrictions on the usage of the data. If they are not accurate enough, undesired situations may appear that can compromise data sovereignty and may include additional delays in data sharing. The former jeopardizes the adoption of DUC. The latter may not be assumed in specific scenarios. To enable the adoption of DUC solutions, these issues must be avoided. This thesis design and develops an approach to analyse the quality of DUC policies. This approach has been validated by implementing it in an algorithm integrated in a DUC solution deployed in a real wind energy use case. It detects all the issues that do not satisfy the policy quality requirements. Furthermore, it provides a performance that is acceptable for an operational environment. Therefore, it will enable the improvement of existing DUC solutions.

Resumen

La cantidad de datos que se generan en todo el mundo crece cada año. Al mismo tiempo, las tecnologías basadas en datos están permitiendo a las organizaciones mejorar sus actividades. Estas tecnologías requieren cada vez más datos para seguir siendo competitivas, especialmente de otras organizaciones. Esto ha convertido los datos y su intercambio entre organizaciones en un recurso clave para el crecimiento económico, la competitividad y la innovación. Sin embargo, un problema crítico que está impidiendo el intercambio de datos en todo el mundo es la reticencia a compartirlos si no se concede la autodeterminación sobre el uso de los mismos. Es lo que se denomina el problema de la soberanía de los datos.

En la actualidad, iniciativas entre las que destaca la Asociación Internacional de Espacios de Datos (IDSA) trabajan en la definición y diseño de una infraestructura distribuida, abierta, interoperable y soberana de servicios en la que diferentes organizaciones puedan colaborar y beneficiarse del intercambio de datos. Para garantizar la soberanía de los datos, IDSA propone el Control de Uso Distribuido (DUC), una particularización del Control de Uso (UC) para escenarios de compartición de datos que amplía el Control de Acceso (AC) para controlar lo que debe ocurrir con los datos a lo largo de su ciclo de vida.

Esta tesis pretende reforzar el intercambio de datos mediante avances en la soberania del dato proporcionando nuevas herramientas para aumentar la adopción de soluciones DUC. En concreto, esta tesis aporta un enfoque novedoso para la identificación y evaluación de las características que deben soportar las soluciones DUC para garantizar la soberanía de los datos. A su vez, esta tesis proporciona mejoras en la calidad de las políticas definidas para restringir el uso de los datos en las soluciones DUC.

En primer lugar, ya existen diferentes soluciones DUC. Sin embargo, al tener objetivos diferentes, presentan características diferentes y específicas. Esto dificulta la identificación de las características que las soluciones DUC deben soportar para garantizar la soberanía de los datos. A su vez, no existe un marco para evaluar en qué medida las soluciones de DUC existentes garantizan la soberanía de los datos. Esta tesis presenta un marco para identificar y evaluar las características que las soluciones DUC deben soportar para cumplir los requisitos de soberanía de datos en el contexto de la puesta en común de datos. Este marco debería suponer un gran paso hacia la adopción de soluciones DUC. Marca el camino para garantizar la soberanía de los datos a través de DUC. Además, permite identificar los puntos fuertes y débiles de las soluciones de DUC en relación con la soberanía de los datos. Este marco se ha validado aplicándolo a las soluciones de DUC más extendidas. Como resultado, se han identificado sus capacidades y limitaciones. Esto permitirá a los proveedores de servicios seleccionar y desplegar la solución más adecuada para un escenario concreto. Además, permitirá a los proveedores de software identificar los puntos débiles de estas soluciones DUC para mejorarlas.

En segundo lugar, una limitación común identificada en todas las soluciones DUC existentes está relacionada con la calidad de las políticas que definen las restricciones de uso de los datos. Si no son lo suficientemente precisas, pueden aparecer situaciones no deseadas que pueden comprometer la soberanía de los datos e incluir retrasos adicionales en su puesta en común. Lo primero pone en peligro la adopción del DUC. Lo segundo puede no asumirse en escenarios específicos. Para permitir la adopción de soluciones DUC, es preciso evitar estos problemas. Esta tesis diseña y desarrolla un enfoque para analizar la calidad de las políticas DUC. Este enfoque se ha validado implementándolo en un algoritmo integrado en una solución DUC desplegada en un caso de uso real de energía eólica. Detecta todos los problemas que no satisfacen los requisitos de calidad de las políticas. Además, proporciona un rendimiento aceptable para un entorno operativo. Por tanto, permitirá mejorar las soluciones DUC existentes.

Contents

List of Figures

List of Tables

List of Abbreviations

ABAC Attribute-Based Access Control

AC Access Control

ADF Attribute Decision Function

BPA Basic Policy Analysis

CAPA Context-Aware Policy Analysis Algorithm

DAC Discretionary Access Control

DMP Distributed Management Point

DRM Digital Rights Management

DUC Distributed Usage Control

EC European Comission

GDP Gross Domestic Product

HTTP Hypertext Transfer Protocol

IDC International Data Corporation

IDSA International Data Spaces Association

IM Information Model

IoT Internet of Things

IT Information Technology

ITU-T International Telecommunication Union - Telecommunication Standardization Sector

LDAP Lightweight Directory Access Protocol

LUCON Label-based Usage CONtrol

MAC Mandatory Access Control

ODF Obligation Decision Function

ODRL Open Digital Rights Language

OEM Original Equipment Manufacturer

OSL Obligation Specification Language

PAP Policy Administration Point

PDP Policy Decision Point

PEP Policy Enforcement Point

PIP Policy Information Point

PMP Policy Management Point

PPL Primelife Policy Language

PRP Policy Retrieval Point

PT Policy Tree

PXP Policy eXecution Point

RAM Reference Architecture Model

RBAC Role-Based Access Control

RT Resource Tree

SAT Boolean Satisfiability Problem

SMP Session Management Point

SoA State of the Art

UC Usage Control

UCON Usage CONtrol

UPL Usage Policy Language

URI Uniform Resource Identifier

W3C World Wide Web Consortium

Chapter 1

Introduction

The International Data Corporation (IDC) defines the Global Datasphere as the combination of data generated, captured or replicated around the world. According to the IDC report on digitization in the world (Reinsel, Gantz, & Rydning, 2018), the Global Datasphere is growing rapidly year by year. In particular, as represented in Figure 1.1, the IDC predicts that the Global Datasphere will grow from 33 zettabytes in 2018 to an expected 175 zettabytes in 2025. This is because traditional data sources are now enriched with a variety of novel sources among which the Internet of Things (IoT) devices stand out.

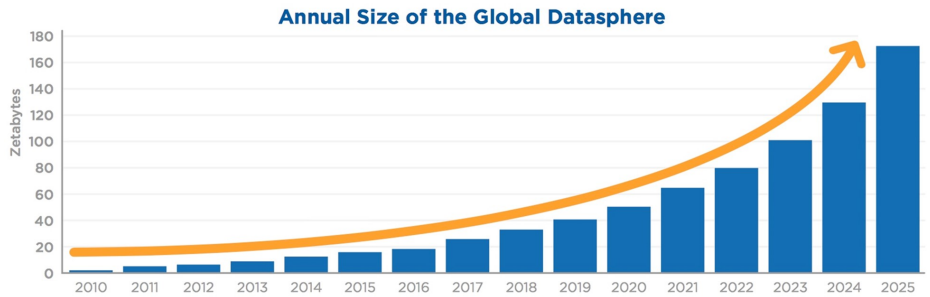

Figure 1.1: Estimated growth of the Global Datasphere.

Simultaneously, data-driven technologies such as data analytics or artificial intelligence have emerged to improve the activities of the organizations. To remain them competitive, they need to use increasingly more data from internal or public data sources, but also, from other organizations. This boosts interest in making the large amount of data generated by organizations a new economic good (Demchenko, de laat, & Los, 2018).

The study carried out by the European Comission (EC) (Scaria, Berghmans, Pont, Arnaut, & Leconte, 2018) confirms that organizations, regardless of their size and sector, are already involved in data sharing or expect to do so in the near future. The organizations that were interviewed recognize the value that data sharing has for them. While data providers identify benefits such as additional revenues, data consumers highlight the increasing efficiency and potential for the development of their products or services.

In particular, the study carried out by IDC on European Data Market (Glennon et al., 2021) establishes that, if favourable policy and legislative conditions are put in place in time and investments in Information Communication Technology (ICT) are encouraged, as represented in Figure 1.2, the value of the European economy around the data may increase from 285 billion €, representing over 1.94% of the European Union Gross Domestic Product (GDP) to 739 billion € by 2020, representing 4% of the overall European Union GDP. As a result, the EC recognizes that great wealth of data and its sharing among organizations is a key resource for the economic growth, competitiveness, and innovation (European Comission, 2020).

This thesis seeks to make contributions that will enable data sharing to meet established expectations. A critical issue that is currently preventing data sharing worldwide is the reluctance of organizations to share their data if the self-determination about its usage is not granted. This is referred to as the data sovereignty concern (Jarke, Otto, & Ram, 2019). In this context, this thesis tries to robust data sharing by focusing on exiting weaknesses around data sovereignty supporting technologies.

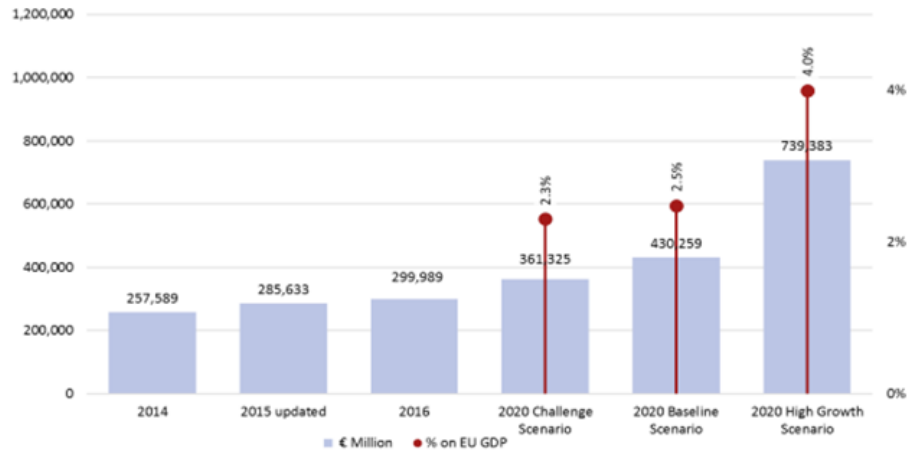

Figure 1.2: Estimated growth of the European data economy.

1.1 Problem and Motivation

In recent years, great efforts are being made to define and design a distributed, open, interoperable, and sovereign infrastructure of services in which different organizations can collaborate and benefit from data sharing. The main initiative in this respect is the International Data Spaces Association (IDSA)[1]. It defines a Reference Architecture Model (RAM) (Otto, Steinbuss, Teuscher, & Lohmann, 2020) for sovereign data sharing between services.

Although data sovereignty is a vital aspect to enable data sharing, existing technologies do not solve this issue completely. This motivates the need for this thesis. The aim of this thesis is to make advances in sovereign data sharing. In this sense, we consider the work carried out in the context of IDSA as the reference. To grant data sovereignty, IDSA proposes Usage Control (UC) (Eitel et al., 2021), an extension of Access Control (AC) for the specification and enforcement of restrictions regulating what must happen to data though its life cycle (Jung & Dörr, 2022). From this general definition of UC,

[1]https://internationaldataspaces.org

this thesis differentiates between UC and Distributed Usage Control (DUC). While UC is limited to a scenario where data is retained and used in an IT system of an organization, DUC addresses those scenarios in which data from one organization is shared and used in an Information Technology (IT) system of another organization. To grant data sovereignty in the context of data sharing we consider DUC as the most appropriate approach. On this basis, the contributions of thesis are made in the context of DUC.

Although some DUC solutions have already been proposed, as they pursue different objectives, they present different, specific features. This makes it hard to determine to what extent they met data sovereignty requirements. Thus, preventing their adoption. Motivated by this concern, the objective of this thesis is to facilitate a widespread adoption of DUC solutions through the identification, assessment and improvement of the features they must support to ensure data sovereignty.

First, it is difficult to identify the features that DUC solutions must support to achieve data sovereignty requirements. Furthermore, there is no framework for the assessment to what extent existing DUC solutions ensure data sovereignty. So, the first objective of the thesis is the construction of a framework to identify and assess the main features that DUC solutions must support to achieve data sovereignty requirements in the context of data sharing. This framework should take a big step towards the adoption of DUC solutions. On the one hand, it may set the path to ensure data sovereignty through DUC. On the other hand, it will enable the identification of strengths and weaknesses of DUC solutions regarding data sovereignty. As a result, the most suitable solution for a particular scenario can be selected. In turn, those existing limitations that may jeopardize the adoption of DUC solutions can be addressed.

Second, the policies that restrict the usage of the data in the context of DUC, if they are not of good quality, can lead to undesired situations that compromise data sovereignty and include additional delays in data sharing. The former prevents the general adoption of

DUC solutions, while the latter may render the applicability of DUC solutions not valid for certain scenarios. To enable the adoption of DUC solutions, these issues must be avoided. However, they have not yet been addressed. As a consequence, the second objective of this thesis is to make improvements on DUC solutions by designing and developing a policy analysis approach to analyse the quality of policies in the context of DUC.

To achieve these objectives, and following a logical approach, this thesis assumes that existing research on AC and UC should be taken as the basis. For instance, to build the framework, we have considered appropriate to identify how data sovereignty requirements in the context of data sharing can be addressed considering the features of existing solutions for AC, UC and DUC itself. Likewise, we consider that the definition of accurate policies for DUC should be addressed based on the AC research literature related to the polcy quality.

1.2 Thesis Objectives and Contributions

The main objective of this thesis is to make advances in sovereign data sharing through the identification and assessment of the main features that DUC solutions must provide and the improvement of the policy quality. To achieve this goal, the following partial objectives have been defined:

- Analyse existing reference architectures for data sharing. Particular attention is given to the approaches followed to ensure data sovereignty.
- Conduct a detailed State of the Art (SoA) of the research related to data control from AC to UC and DUC.
- From that SoA, build a framework for the identification and assessment of the features that DUC solutions must support to met data sovereignty requirements in the context of data sharing.

- Validation of the framework built applying it to the assessment of the most widespread DUC solutions.

- Conduct a detailed SoA of the research related to the policy quality in the context of AC.

- Considering that SoA, design and develop a policy analysis approach to analyse the policy quality in different DUC solutions.

- Validation of the proposed solution of policy analysis. This requires the deployment of a real use case and the development of a test plan.

As a result of the achievement of the goals described above, this thesis makes the following contributions:

- A detailed review of existing research related to data control from AC to UC and DUC. This contribution is addressed in Chapter 2.

- A framework, considering data sovereignty requirements in the context of data sharing and based on existing research related to AC, UC, and DUC, for the identification and assessment of the features that must be supported by DUC solutions to ensure data sovereignty. This contribution is addressed in Chapter 3.

- Applying the proposed framework, the identification of capabilities and limitations of existing DUC solutions to ensure data sovereignty. This contribution is addressed in Chapter 5.

- An in-depth review of existing research to analyse the policy quality for AC policies. This contribution is addressed in Chapter 2.

- A policy analysis approach, based on the AC research literature, to address the policy quality for DUC policies. This contribution is addressed in Chapter 4.

- An algorithm, based on the proposed policy analysis approach, for the analysis of the quality of DUC policies. This contribution is addressed in Section 4.

1.3 Thesis Structure

This thesis is divided in the following chapters:

- Chapter 2: Related Work. This chapter describes existing research related to the contributions of this thesis. It addresses the technological context of DUC from AC to UC and DUC itself, existing approaches proposed to address DUC and existing research related to the assessment of AC and DUC solutions and the policy quality in AC and DUC.

- Chapter 3: Framework for the Identification and Assessment of Main Features of Distributed Usage Control Solutions. This chapter, considering data sovereignty requirements in the context of data sharing and based on the analysis of the research related to AC, UC and DUC, describes the framework to identify and assess the features that must be supported by DUC solutions to ensure data sovereignty.

- Chapter 4: Context-Aware Policy Analysis Algorithm for Distributed Usage Control. This chapter, considering the AC research literature, describes the approach proposed to analyse the policy quality for DUC policies. Based on this approach, it presents a policy analysis algorithm to analyse the quality of DUC policies.

- Chapter 5: Validation. This chapter validates the contributions of this thesis. It applies the framework for the assessment of DUC solutions to the most widespread DUC solutions. As a result, their capabilities and limitations are identified. Also, it describes the implementation of the policy analysis algorithm

for a real wind energy use case, presents a test plan and analyses whether it ensures the policy quality the performance provided to do so.

- Chapter 6: Conclusions and Further Work. This chapter includes the main contributions of this thesis and future lines of research are discussed.

1.4 Publications

The contributions of this thesis have been presented and published in different conferences and scientific journals of impact. In this section all this work is listed and linked to the chapters of this thesis to which they are related.

1.4.1 Conference Papers

- Gil, G., Arnaiz, A., & Higuero, M. (2019). Theoretical Assessment of existing frameworks for data usage control: Strength and limitations with respect to current application scenarios. In *Proceedings of the 9th International Conference on the Internet of Things*. Retrieved from `https://doi.org/10.1145/3365871.3365896`. **Chapter 3: Framework for the Identification and Assessment of Main Features of Distributed Usage Control Solutions**.

- Gil, G. & Esnaola-Gonzalez, I. (2020). Towards defining Data Usage Restrictions in the Built Environment. In *Proceedings of the 8th Linked Data in Architecture and Construction (LDAC) Workshop* (pp. 37-49). Retrieved from `http://ceur-ws.org/Vol-2636/03paper.pdf`. **Chapter 4: Context-Aware Policy Analysis Algorithm for Distributed Usage Control**.

- Gil, G., Arnaiz, A., Higuero, M. & Diez, F.J. (2020). Evaluation Methodology for Distributed Data Usage Control Solu-

tions. In *Global Internet of Things Summit (GIoTS)* (pp. 1-6). Retrieved from https://doi.org/10.1109/GIOTS49054.2020.9119565. **Chapter 3: Framework for the Identification and Assessment of Main Features of Distributed Usage Control Solutions**.

- Gil, G. & Esnaola-Gonzalez, I. (2022). Semantic Interoperability, a Key enabler for Good Quality Distributed Usage Control. In *Workshop: Data Spaces & Semantic Interoperability.* Retrieved from https://www.trusts-data.eu/wp-content/uploads/2022/08/Semantic-Interoperability-a-Key-Enabler-for-Good-Quality-Usage-Control.pdf. **Chapter 4: Context-Aware Policy Analysis Algorithm for Distributed Usage Control**.

1.4.2 Journal Publications

- Gil, G., Arnaiz, A., Higuero, M. & Diez, F.J. (2022). Assessment Framework for the Identification and Evaluation of Main Features for Distributed Usage Control Solutions. *ACM Transactions on Privacy and Security, 26*(1), 1-28. Retrieved from https://doi.org/10.1145/3561511. **Chapter 3: Framework for the Identification and Assessment of Main Features of Distributed Usage Control Solutions and Chapter 5: Validation**.

- Gil, G., Arnaiz, A., Higuero, M., Diez, F.J. & Jacob, E. (2022). Context-Aware Policy Analysis for Distributed Usage Control. *Energies, 15*(19), 7113. Retrieved from https://doi.org/10.3390/en15197113. **Chapter 4: Context-Aware Policy Analysis Algorithm for Distributed Usage Control and Chapter 5: Validation**.

Chapter 2

Related Work

2.1 Introduction

This chapter includes a comprehensive review of the existing literature related to the contributions of this thesis. In particular, the technological context of DUC is presented. Afterwards, the approaches proposed to address DUC are interpreted. In addition, the assessment of DUC solutions is analysed. Finally, research related to the quality of the policies for DUC is examined.

2.2 Technological Context

The International Telecommunication Union - Telecommunication Standardization Sector (ITU-T) X.800 recommendation defines AC as the prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner (ITU-T, 1991). In this definition, the use of a resource refers to the access to it, while the manner of use refers to the purpose of the access, such as read or write. The concept of resource has a broad scope. This thesis focuses on data as a resource.

From this definition we can deduce that if data is accessed, AC does not deal with how it is used afterwards. In this context, UC is defined as an extension of AC for the specification and enforcement of restrictions regulating what must happen to data through its life cycle (Jung & Dörr, 2022). From the general definition of UC, this thesis differentiates between UC and DUC based on the following two scenarios:

- Centralized architecture: Data is retained and used in a IT system of an organization. The scope of UC is limited to this scenario.

- Distributed architecture: Data from an organization is shared and used in a IT system of another organization. DUC extends UC to address this scenario.

This thesis aims to encourage data sharing among organizations to enable the benefits it presents. For this purpose, data sovereignty is considered of utmost importance. Therefore, the self-determination of individuals and organizations regarding the usage of their data must be ensured. To grant data sovereignty in data sharing scenarios, we consider that DUC is the most suitable approach. Therefore, the contributions of this thesis are focused on the DUC context.

Considering AC the basis for UC and, in turn, UC the basis for DUC, we consider of interest to describe in this section the most significant aspects of AC, UC and DUC.

2.2.1 Access Control

AC prevents the unauthorized access to the data. To do so, it controls restrictions on the access to the data. However, once data is accessed, it is out of the scope of AC to control how it is used afterwards.

The main elements in the AC context are subjects, objects and rights. These are described below:

- Subjects: These are the entities that can perform actions of access to data. Examples of subjects are an individual or a software component.
- Objects: Those entities that represent resources to which access need to be controlled. In this thesis we focus on data as a resource and, consequently as an object.
- Rights: The actions performed by subjects on objects. In data access, typical examples are read and write.

Based on these elements and motivated by the different requirements to restrict the access to the data, different strategies have been developed for implementing AC. As a result, different AC models have been proposed (Penelova, 2021), some simple ones, among which Role-Based Access Control (RBAC), Mandatory Access Control (MAC) and Discretionary Access Control (DAC) stand out and others more complex such as Attribute-Based Access Control (ABAC). These are described below.

- The RBAC model (Sandhu, 1998): The basic concept of this model is that rights to access to objects are associated with roles, and subjects are made members of appropriate roles, thereby acquiring the rights of the roles to access objects. It is an efficient approach to manage a large set of subjects based on role membership.
- The MAC model (United States Department of Defense, 1985): As defined by the Trusted Computer System Evaluation Criteria, this model controls the access to the objects based on the sensitivity of the objects represented by labels (i.e., security label) and the rights of subjects to access objects of that sensitivity (i.e, clearance levels). It provides a very high level of security. However, its management is complex.

- The DAC model (United States Department of Defense, 1985): As defined by the The Trusted Computer System Evaluation Criteria, this model sets owners for the objects to be controlled. Owners define access control lists for objects. These lists contain the rights that subjects have over the corresponding object. This model is discretionary in the sense that a subject with specific rights on an object can pass these rights to another subject. It is a flexible approach to control the access to each object based on an access control list. However it is a challenging approach to manage complex systems as there is not a centralized access management.

- The ABAC model (Chung et al., 2019): It is defined as an access control method in which the requests from subjects for specific rights on objects are granted or denied based on the attributes of the subjects, objects, environment conditions, and a set of policies that are specified in terms of these attributes. Attributes describe properties about subjects, objects and the environment. Examples of attributes are the role of a subject, the identifier of an object or the current environment time or location. Policies restrict the access to the data based on the values of the attributes of the subject, object and environment. It is a flexible approach that can implement AC considering the richness of the attributes, making it ideal for rapidly changing environments (Hu, Kuhn, Ferraiolo, & Voas, 2015).

From all existing AC models, the ABAC model is established as the basis to address UC (Sandhu & Park, 2003). We consider that this is a proper approach mainly due to two main reasons:

- Flexibility: The use of attributes allows to define complex restrictions. This makes it possible to cover the requirements to restrict the usage of the data in the UC context.

- Context-awareness: The use of attributes allows to control the data in a changing environment. This makes it possible to control data throughout its life cycle.

Since the ABAC model has been used as the basis of UC, in this work we consider interesting to describe it below. The components used in the ABAC model to restrict the access to the data are identified. Furthermore, the architecture proposed to control the access to the data is presented.

The ABAC model restricts the access to the data based on authorizations (A). Authorizations define whether subjects (S) can access objects (O) based on the requested rights (R) and the attributes of the subjects (ATTR(S)), the objects (ATTR(O)) and the environment (ATTR(E)). Figure 2.1 represents the ABAC model components and the relationships among them.

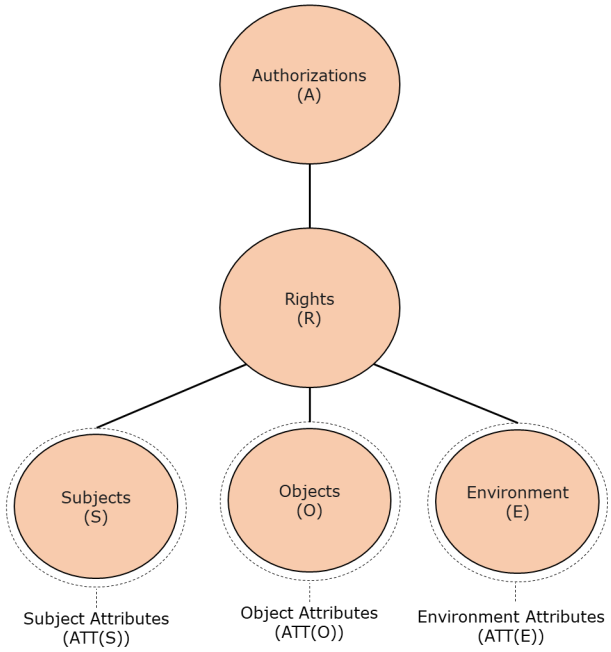

Figure 2.1: The ABAC model components.

To control data access, the ABAC model proposes an architecture composed of several components: These are the Policy Administration Point (PAP), the Policy Enforcement Point (PEP), the Policy Decision Point (PDP) and the Policy Information Point (PIP). Fig-

ure 2.2 represents it. The functions of each component are described below.

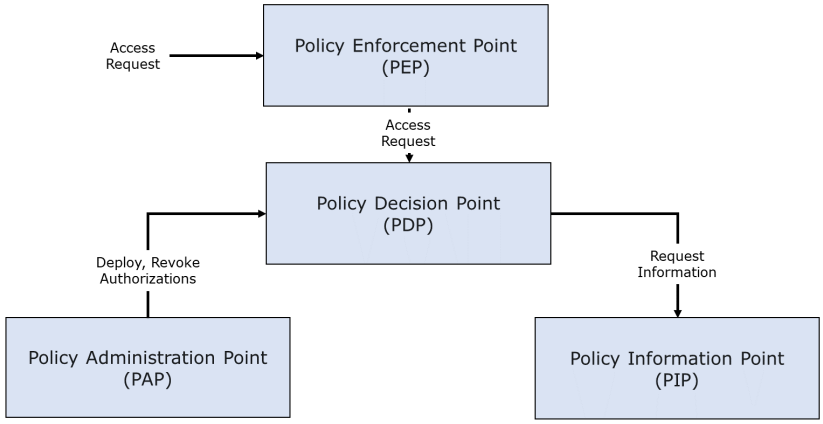

Figure 2.2: The ABAC model architecture.

- PAP: It provides the interfaces to deploy and revoke authorizations in the PDP.
- PEP: It intercepts access requests from subjects to objects and forwards them to the PDP.
- PDP: It makes access decisions. To do so, it evaluates authorizations. Based on the requested rights it evaluates the attributes of subjects, objects and the environment. These attributes are requested to the PIP.
- PIP: It stores the attributes of the subjects, objects and the environment.

Therefore, authorizations are initially deployed in the PDP through the PAP. Each time an access request is intercepted by the PEP, it is forwarded to the PDP. In this moment, those authorizations that apply to the intercepted access request are evaluated by the PDP.

The information required to do so is requested by the PDP to the PIP. As a result of the evaluation, the PDP responds to the PEP with an access decision. Based on the response, the PDP grants or denies the access to the data.

2.2.2 Usage Control

As mentioned above, AC prevents the unauthorized access to the data. To do so, it controls restrictions on the access to the data that are concerned with whether the data may be released in the first place. However, it does not control how data is used afterwards. UC goes a step further and controls how data is used through its life cycle. In this regard, it extends AC to control restrictions that govern the future usage of the data regarding the moment of access to it. This is represented in Figure 2.3.

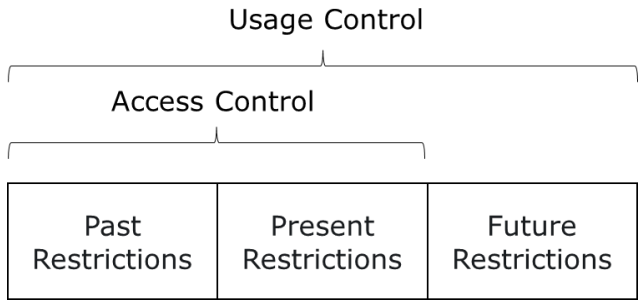

Figure 2.3: Usage control.

As initially stated, from the general definition of UC (Jung & Dörr, 2022), in this thesis we limit the scope of UC to a centralized architecture. This said, UC is addressed based on the ABAC model (Sandhu & Park, 2003). As previously highlighted, we consider that this is the most adequate strategy because the use of attributes enables to define complex restrictions to cover the requirements to restrict the usage of the data in the UC context and to control data in a changing environment such as the data life cycle.

Based on the ABAC model, research in the field of UC led to the development of the seminal Usage CONtrol (UCON) model (Park & Sandhu, 2004) and, later, to its extension (Katt, Zhang, Breu, Hafner, & Seifert, 2008). The UCON model has been considered the reference framework to implement UC (Aliaksandr, Fabio, & Paolo, 2010). Furthermore, based on it, several approaches have been proposed to address DUC. Therefore, we consider interesting to describe below the most relevant aspects of the UCON model.

The UCON model restricts data usage based on authorizations (A), conditions (C) and obligations (B). While authorizations are already considered in the ABAC model, conditions and obligations are new concepts introduced. Figure 2.4 represents the UCON model components and the relationships among them. In addition, these components and their functions are described below.

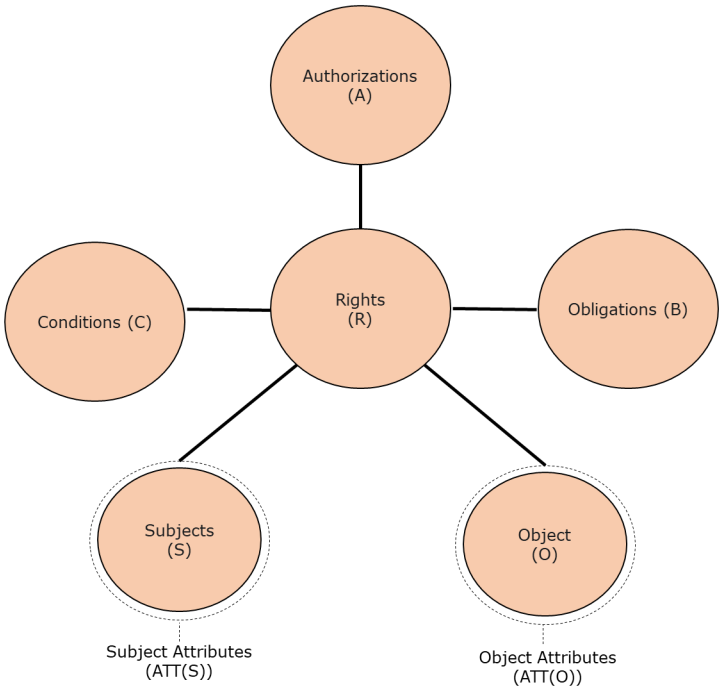

Figure 2.4: The UCON model components.

- Authorizations: They define if subjects can use objects based on the requested rights (R) and the attributes of the subjects (ATTR(S)) and the objects (ATTR(O)).
- Conditions: They are concerned with whether data usage is granted or not based on environment attributes. It does covers those environment attributes considered in authorizations for the ABAC model.
- Obligations: It is a new concept introduced by the UCON model. They are divided into subject obligations and system obligations.
    - Subject obligations: They are actions that must be performed by the subject. An example is to accept a license agreement.
    - System obligations: They are actions that must be performed by the system. Examples are logging or data deletion.

With the aim of controlling data from the access moment along a period of time when data is used several times, the UCON model introduces two new concepts with respect to the ABAC model. Those are decision continuity and obligation handling. They are described below:

- Decision continuity: It refers to the continuous enforcement of authorizations, conditions and obligations along the time when data is used multiple times. Similar to the ABAC model, the UCON model controls the access to data. Therefore, it enforces authorizations, obligations and conditions each time data is accessed. However, once access to data is granted, subject, object and environment attributes can change. This is referred to as attribute mutability (Park, Zhang, & Sandhu, 2004). In this regard, the UCON model monitors and intercepts attribute updates to consequently trigger the enforcement of authorizations, conditions and obligations.

- Obligation handling: It ensures that system obligations are being fulfilled. Several system obligations must be performed each time data is accessed. Also, due to decision continuity, some system obligations must be executed after the access to the data if subject, object or environment attributes are updated during the period of time that data is being used. In turn, specific system obligations must be carried out once the usage session has ended. This is the case of a data deletion.

Considering these concepts, Figure 2.5 represents the policy enforcement state transition diagram that is implemented in the UCON model to control data from the moment of the first access along the period of time when it is used multiple times. Solid arrows in the figure represent transitions within the state diagram triggered by external agents. These are for example usage requests. Dashed arrows in the figure symbolize transitions triggered by the UCON model. Examples of these are the enforcement of authorizations, conditions and obligations and the corresponding results.

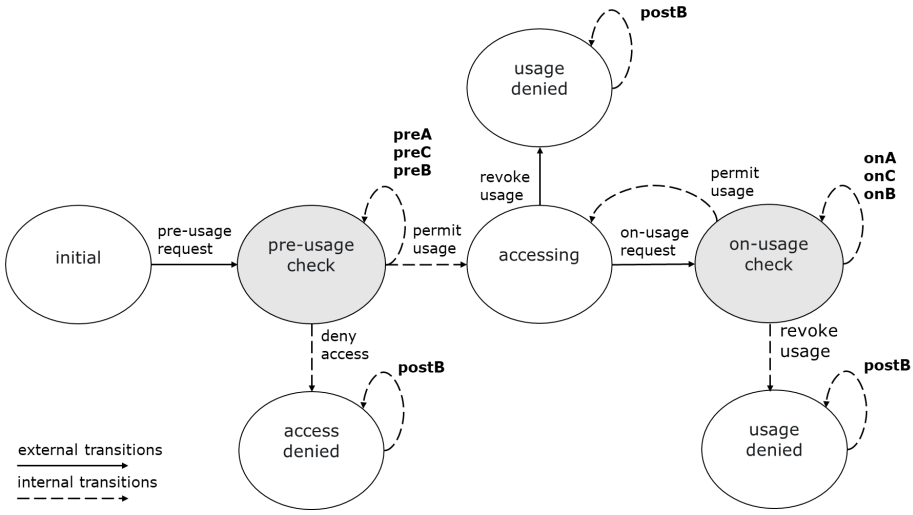

Figure 2.5: The UCON model policy enforcement state transition diagram.

To make a more comprehensive analysis of the state diagram, we describe it by means of explaining three phases: before usage, when access to data has not yet been granted; during usage, when access to data is granted and decision continuity come into play; after usage, when the usage session has ended. These phases are described below.

- Before usage: Every time data access is requested, the pre-usage request is triggered, and authorizations, obligations and conditions are enforced at the pre-usage check state. These enforcements are respectively referred to as preA, preB and preC. So far, there is no difference with respect to the ABAC model.

- During usage: If access to the data is granted, the permit usage is triggered and the accessing state is reached. In this state, data can be used multiple times. Every time an attribute update is detected, an on-usage request is triggered and authorizations, conditions and obligations are enforced at the on-usage check state. These are respectively referred to as onA, onB and onC. Thus, decision continuity is deployed. If authorizations, conditions and obligations are satisfied, data usage is granted, permit usage is triggered and the accessing state reached again.

- After usage: If data access or usage is respectively denied at the pre-usage check or on-usage check state or the usage ended by an external event, postB are triggered and their enforcement tracked.

To control data usage in a centralized architecture, the UCON model proposes an architecture that is considered as an extension of the ABAC model architecture. This architecture is composed of the PAP, the PEP, the Session Management Point (SMP), the PDP and the PIP components. With respect the ABAC model architecture, all the components are included. However, only the scope of the PEP and the PIP remain the same. In turn, a new component called SMP has been introduced to address the new features of decision

continuity and obligation handling. Figure 2.6 represents it. The main functions of each component are described below.

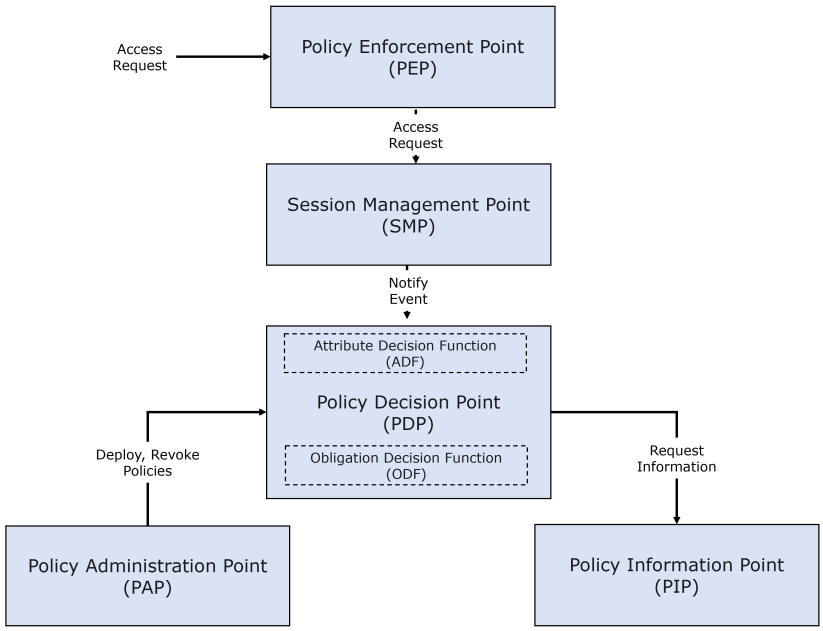

Figure 2.6: The UCON model architecture.

- PAP: In the ABAC model, it provides the interfaces to deploy and revoke authorizations. In the UCON model, these interfaces are extended to deploy conditions and obligations too.

- PEP: It intercepts access requests from subjects to objects and forwards them to the SMP instead of the PDP, as happens in the ABAC model. Thus, decision continuity and obligation handling will be managed.

- SMP: It manages the state transition diagram shown in Figure 2.5. This component is responsible for supporting decision continuity and obligation handling. How it does so is described below.

    - Decision continuity: When the access request is received at the SMP from the PEP, the SMP triggers the enforcement of authorizations and conditions and obligations to the PDP. In turn, if access to data is granted, the SMP monitors attribute updates to trigger the enforcement of authorizations, conditions and obligations to the PDP in the period that data is used multiple times.

    - Obligation handling: The SMP triggers the fulfillment of system obligations to the PDP. Due to decision continuity, several system obligations are triggered before the usage session when the access request is received and others during the usage session when an attribute update is intercepted. Furthermore, some obligations are triggered after the usage session either because the access request is denied, the usage session revoked or the session ended by the subject.

- PDP: Based on the requested rights, it enforces authorizations, conditions and obligations. It is composed of a Attribute Decision Function (ADF) and a Obligation Decision Function (ODF).

    - ADF: It enforces authorizations and conditions. To this end, it evaluates the attributes of subjects, objects and the environment. Attributes are requested to the PIP. Therefore, it performs the same functionalities as the PDP in the ABAC model.

    - ODF: It extends the scope of the PDP in the ABAC model to evaluate subject obligations and enforced system obligations.

- PIP: It stores the attributes of the subjects, objects and the environment.

2.2.3 Distributed Usage Control

In this thesis, as initially remarked, we differentiate between UC and DUC. Both control the data through its life cycle. Nevertheless, while UC address it for a centralized architecture, DUC extends UC to address it for a distributed architecture.

At present, several DUC solutions have been developed following different approaches. The following section provides a detailed analysis of the approaches defined for DUC so far. We consider this review of great interest as the technological starting point to identify the features that DUC solutions must support to met the requirements to ensure data sovereignty in the context of data sharing.

2.3 Analysis of DUC approaches

To analyse the approaches proposed so far for DUC, we observe that there are two key aspects to be examined. These are described below:

- DUC policy languages: These languages are specifications used in DUC approaches to express policies that articulate the usage of the data in a distributed architecture. Different DUC policy languages have been defined. In this regard, it should be considered that the capabilities and complexity of the languages have evolved throughout the research literature as new requirements related to data usage have been included in the DUC context.

- DUC architectures: They refer to the set of components included in DUC approaches to control data usage in a distributed architecture. Each approach proposes to address DUC differently. Thus, they present different architectures.

On this basis, to make a comprehensive analysis of DUC approaches, this section analyses the evolution of the policy languages in DUC and reviews the architectures defined to provide a solution for DUC.

2.3.1 DUC Policy Languages

DUC approaches make use of different DUC policy languages to express how the data can be used in a distributed architecture. These DUC policy languages have followed an incremental approach throughout the research literature refining the concepts used previously and increasing their scope. This section reviews the concepts that each of the DUC policy languages introduces and analyses how they have evolved.

2.3.1.1 Primelife Policy Language

The Primelife Policy Language (PPL) (Ardagna et al., 2009) is a language developed to express policies that define privacy-preserving restrictions on the usage of the data once it has been shared. It is based on the concepts introduced in the ABAC model for restricting the access to the data. It can express authorizations, conditions, provisional actions and obligations. These concepts are described below:

- Authorizations: They define the rights that subjects have to use objects taking into account the attributes of subjects and objects. With respect to the ABAC model, authorizations are extended to include refinements. An example of refinement is the authorization purpose. However, authorizations do not consider environment attributes.

- Conditions: These define further restrictions for subjects to use objects considering environment attributes. It does cover

the environment attributes considered in authorizations of the ABAC model.

- Provisional actions: They are actions that must be executed by the subject to grant data usage.

- Obligations: They are actions that must be executed by the system triggered by different events such as the usage request itself or a time period.

We identify two main problems in this policy language. Restrictions on data usage does not consider how data is shared. Furthermore, there is a lack of expressiveness of the concepts introduced. Only examples for specific privacy-preserving restrictions on data usage are addressed.

2.3.1.2 Obligation Specification Language

The Obligation Specification Language (OSL) (Hilty, Pretschner, Basin, Schaefer, & Walter, 2007) emerges to resolve the lack of a general-purpose policy language that supports many usage restrictions demanded in that moment. In the context of DUC, two actors are involved: the data provider who distributes the data and the data consumer who requests and receives the data for its usage. Considering this, OSL is based on the concepts of provisions and obligations (Bettini, Jajodia, Wang, & Wijesekera, 2003). They are described below.

- Provisions: These are those conditions that refer to whether the data may be released by the data provider. These conditions include authorizations, conditions and provisional actions supported by the PPL described above.

- Obligations: These are conditions that govern the usage of the data in the data consumer. Obligations are divided into usage restrictions and actions requirements. These concepts are described below:

- Usage restrictions: They prohibit certain usages under given conditions. These conditions include authorizations, conditions and provisional actions included in the PPL.
- Action requirements: They express mandatory actions that must be executed under specific conditions. These conditions include the moment when the action must be executed and how. These action requirements include obligations considered in the PPL.

As a result, we conclude that OSL goes one step further than PPL and considers restrictions on the disclosure of the data apart from those already contemplated in PPL for the usage of the data once it has been shared. In this context, the concepts of provisions and obligations from the OSL cover most of the concepts introduced in the PPL. The main limitation in OSL is that obligations from the PPL are not included within provisions. Thus, system actions like data anonymization can not be performed before data is released. The main advantage of OSL over PPL is that it enables the expression of multiple different restrictions on data usage.

2.3.1.3 Open Digital Rights Language

The World Wide Web Consortium (W3C) subsequently recommended the Open Digital Rights Language (ODRL) (Iannella, 2018). It refines the concepts previously defined and increases their scope through a policy language that provides an interoperable vocabulary and a data model for the description of statements about the usage of data. The ODRL includes permissions or prohibitions and duties. These are described below:

- Permissions and prohibitions: They describe if an action related to data usage is permitted or prohibited to perform on a resource under certain conditions. These actions can be the initial usage request in the data provider or further usage requests in the data consumer. Thus, permissions and provisions

include provisions and usage restrictions previously defined in the OSL language.

- Duties: They are associated to a permission and describe the actions that must be performed if the permission is satisfied. Thus, duties include action requirements from OSL not only related to obligations in the data consumer, but also, provisions in the data provider.

Therefore, we can conclude that the ODRL includes all the concepts introduced in OSL and extends it to include action requirements also in the context of provisions. In turn, the expressiveness it provides for permissions, prohibitions and duties enables the definition of a wide variety of usage restrictions.

2.3.1.4 IDSA Usage Policy Language

In an study carried out to identify those usage restrictions that were required in different projects where the IDSA was involved[1], a lack of expressiveness and ambiguity in the ODRL was identified. As a result, the IDSA Usage Policy Language (UPL) was developed. The IDSA UPL maintains the general structure of the ODRL. That is, it includes permissions, prohibitions and duties. It increases the expressiveness of these concepts. Thus, it enables the expression of more diverse usage restrictions.

We consider that a DUC policy language should consider two aspects to reflect restrictions related to data usage. These are permissions and prohibitions that define the conditions under which a usage request is permitted or prohibited and duties that set actions that must be performed as a result of a data permission. On this basis, we consider the reference DUC policy language the one that enables the definition of much diverse conditions related to permissions and prohibitions and actions related to duties. Based on these

[1]https://internationaldataspaces.org/make/projects/

considerations, we think that the IDSA UPL is the the reference policy language for DUC. Therefore, we consider of interest to describe below the main components of the IDSA UPL. Figure 2.7 represents its general architecture.

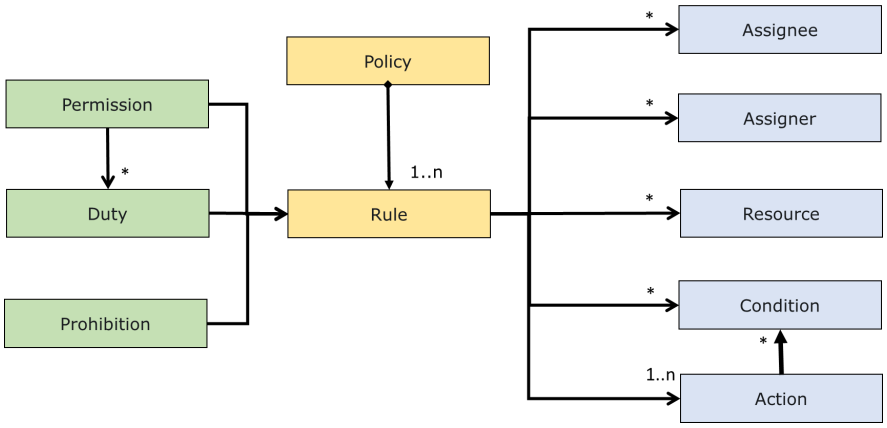

Figure 2.7: IDSA UPL general structure.

In the IDSA UPL a policy must be composed of a set of rules. As shown on the left side of the figure, there are three type of rules: permission, prohibition and duty. They are describe below:

- Permission or prohibition: It allows or prohibit a data usage action to be exercised on an resource if all conditions are satisfied.
- Duty: The obligation to exercise an action with all the conditions satisfied.

As shown in the right part of the figure all the rules are in turn composed of an assignee, an assigner, a resource, a condition and a action. These are describe below:

- Assignee: The recipient of the policy statement.
- Assigner: The issuer of the policy statement.

- Resource: It is a single digital asset or a coherent set of digital contents to be controlled regarding its usage. In this thesis we focus on resource comprising data.

- Condition: It is a boolean/logical expression that can be used to refine the rules. For permission or prohibition rules defines when the rule applies. For example a time period. For a duty rule defines how the rule should be applied. For example, where to perform a logging. It compares two operands with one relational operator. The expressiveness of the operands and the operator enables the definition of a large number of different conditions.

- Action: An act one might be permitted, prohibited to do related to data usage or obligated to perform as a consequence of a usage permission.

2.3.2 DUC Architectures

Several proposals have been developed over the last years to control how data is used in a distributed architecture. Each approach addresses DUC differently. Therefore, they present different architectures. This section reviews the strategies of these approaches, how they address the objectives pursued and their main strengths and limitations.

2.3.2.1 Observation-based Distributed Usage Control

This approach (Pretschner, Hilty, & Basin, 2006) has been developed with the aim of controlling, in service-oriented architectures, how data from service providers is used by service consumers in their own infrastructure, considering that service providers do not have control over the infrastructure of the service consumer.

For this purpose, the service provider defines restrictions related to data usage making use of the OSL. In turn, the service provider controls usage requests from service consumers. If a usage request is intercepted, policies are enforced. If data usage is granted, the data is shared and the service provider detects if policies are being complied or not during and after data usage by the service consumer, based on the new concept of observability (Pretschner, Massacci, & Hilty, 2007). This concept involves the enforcement of policies in the service provider side through the reception and analysis of those signals from the service consumer side that impact on the compliance of policies (e.g. whether data has been deleted after a period of time).

In particular, to remotely monitor policy compliance during and after data usage a set of observational mechanisms are set up for the set of policies defined. However, deploying observational mechanisms for a large set of highly expressive policies such as those defined by means of OSL is not feasible. Also, observational mechanisms do not prevent policy violations rather than detect them. Therefore, this approach is highly prone to security breaches. For this reason, we do not consider this approach suitable for DUC as data sovereignty is compromised which jeopardizes data sharing.

This approach concludes that there is a need to handle policy propagation. That is, service providers must share the data with the policies attached. These policies must be enforced during and after the use of the data by the service consumer.

2.3.2.2 Privacy-preserving Distributed Usage Control

This DUC strategy (Di Cerbo, Some, Gomez, & Trabelsi, 2015) has the goal to control, in cloud to distributed mobile communications, how the data from the cloud is used on distributed mobile devices.

To this end, policies related to data usage are defined in the cloud by means of the PPL. These policies are propagated to mobile devices

following the sticky policy approach (Di Cerbo, Trabelsi, Steingruber, Dodero, & Bezzi, 2013). This means that policies are shared attached to the data. Thus, they are enforced on distributed mobile devices during and after data usage.

For the enforcement of policies on mobile devices, this approach proposes an architecture based on the ABAC architecture (Trabelsi, Sendor, & Reinicke, 2011). Figure 2.8 represents it.

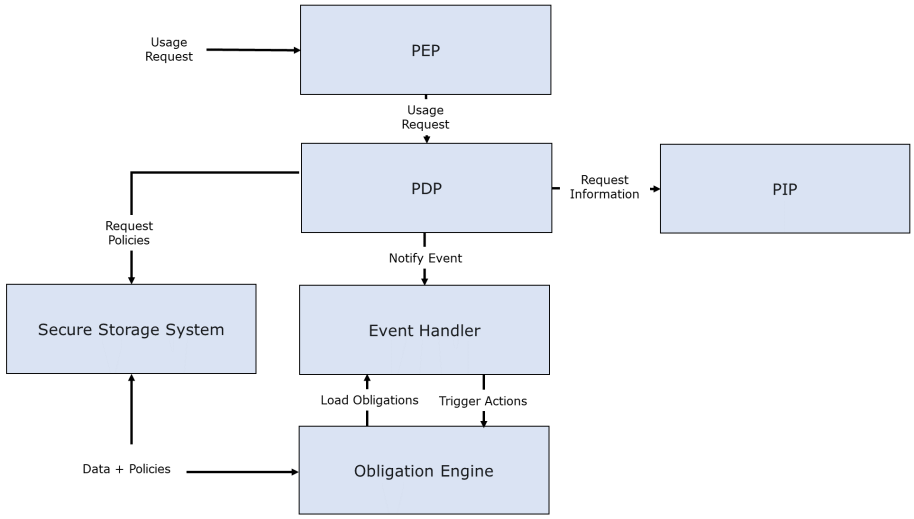

Figure 2.8: Privacy-preserving distributed usage control architecture.

It introduces three new components with respect to the ABAC model architecture. These are the secure storage system, the event handler and the obligation engine. All the components are described below:

- Secure storage system: When data is downloaded from the cloud to a distributed mobile device, it stores the data itself as well as the sticky policies.
- Obligation engine: When data is downloaded from the cloud, it extracts obligations from policies. Obligations are defined as pairs trigger action. In turn, a trigger is an event that can be

filtered by conditions. An example of this are a usage logging or a data deletion obligation. The usage logging is triggered if a usage permission is intercepted without any condition. The action to delete data is triggered too if a usage request permission is intercepted but when a time period is elapsed. On this basis, the obligation engine loads obligations in the event handler, and is responsible for the execution of the actions when the corresponding events are intercepted and the conditions met.

- Event handler: It monitors and detects events and triggers the corresponding actions to the obligation engine when the associate conditions are met. It thus addresses the support of obligation handling.

- PEP: Once data and its policies are downloaded, instead of access requests, as for the ABAC model architecture, it intercepts each of the usage requests and forwards them to the PDP. Thus, it is responsible for the support of decision continuity.

- PDP: It has the following functions:

    - It requests policies from the secure storage system for their enforcement.
    - It evaluates authorizations, conditions and provisional actions from the policies. The scope is extended with respect to the one from the ABAC model architecture to address the new concepts introduced within PPL.
    - As a result of policy enforcement, it triggers the associated event to the event handler. Thus, the corresponding obligation can be performed by the obligation engine.

- PIP: As in the ABAC model architecture, it stores the information requested by the PDP to evaluate policies.

This architecture is the first to address policy propagation. In this sense, we think that policy propagation is the most appropriate approach to be able to control how shared data is used during and

after data usage in a distributed architecture. It should be noted that this architecture is deployed in mobile devices. Thus, it does not control data before usage at the cloud. This is delegated to an Hypertext Transfer Protocol (HTTP) authentication method implemented in the data request. To fully control data through its life cycle, we consider that a DUC architecture should control how data is shared too.

2.3.2.3 Fully Decentralized Distributed Usage Control

Based on their previous work (Pretschner et al., 2006), the fully decentralized distributed usage control approach (Kelbert & Pretschner, 2018) aims to control how the data from a service provider is used by service consumers based on the propagation of policies defined by service providers using the OSL.

To achieve this goal, it identifies and addresses the following three issues:

- Propagation of policies: If a usage request is granted at the service provider, policies must be shared attached to the data to be enforced by service consumers during and after the data life cycle.
- Data flow tracking: Policy enforcement must be performed considering data flow tracking information which is gathered along the distributed architecture.
- Coordination of policies: The enforcement of policies depends on information shared along the architecture. For example, the total number of data usages.

To address these issues, the authors propose an architecture (Kelbert & Pretschner, 2015) based on the ABAC architecture that is deployed both in the service provider and consumer sides. Figure 2.9 represents it.

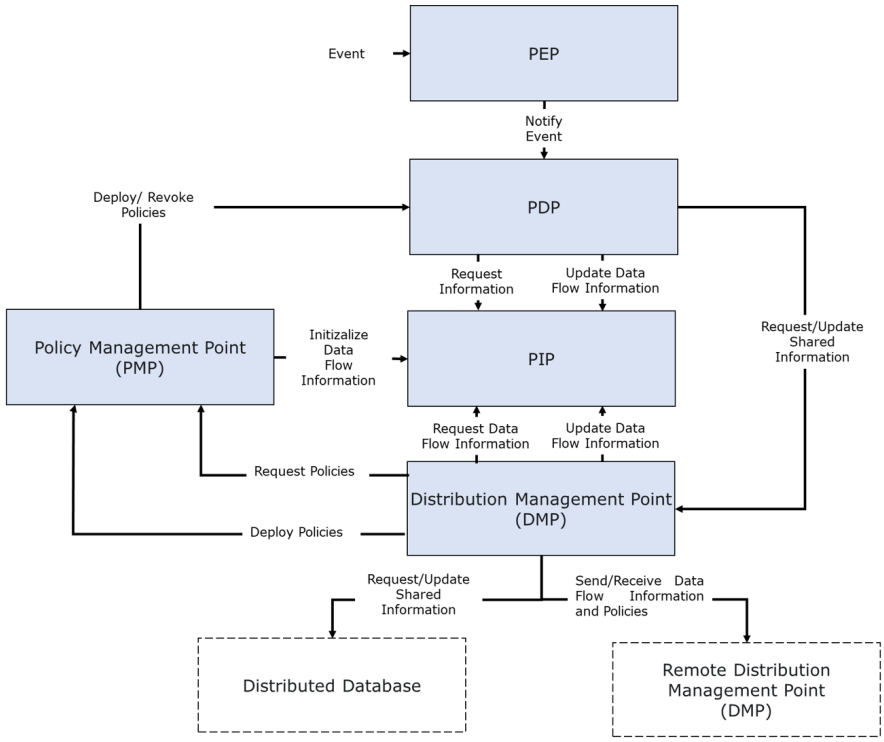

Figure 2.9: Fully decentralized distributed usage control architecture.

With respect to the ABAC architecture, this architecture introduces three new components. These are the Policy Management Point (PMP), the Distributed Management Point (DMP) and the distributed database. The scope of each component is described below.

- PMP: It implements a twofold function:
  - It deploys and revokes policies in the PDP. It is similar to the PAP in the ABAC model architecture.
  - When a policy is deployed in the PDP, if required, it initializes data flow tracking information (simplified as data flow information) in the PIP.

- PEP: It extends the function assigned to the component with the same name in the ABAC model architecture to intercept, not only usage requests at service providers and service consumers, but also, any event that triggers the enforcement of policies during or after data usage such as a time event. Thus, decision continuity and obligation handling are supported. In all cases, policy enforcement is forwarded to the PDP.

- PDP: As in the ABAC model architecture, this component is responsible for the enforcement of policies. This process is more complex in this architecture. Provisions and usage restrictions must be evaluated and action requirements executed. To do so, it interacts with the PIP for two reasons:

    - To request information from the PIP that is required to evaluate provisions and usage restrictions as happens in the ABAC model architecture.
    - To update data flow information in the PIP as a consequence of policy enforcement.

  In turn, it interacts with the DMP with two goals:

    - To request the information shared in the distributed architecture that must be considered for policy enforcement.
    - To notify about updates on relevant information that affect the enforcement of policies in the distributed architecture.

- PIP: This module plays the same role as in the architecture proposed by the ABAC model. It stores the information required by the PDP to evaluate policies.

- DMP: It performs several functions to support the propagation of policies, data flow tracking and coordination of policies along the distributed architecture.

  In the service provider, if access to data is granted by the PDP:

    - It requests the policies related to the data to the PMP.

  - It requests data flow information to the PIP.
  - It sends the policies and data flow information attached to the data to the remote DMP, to be used for policy enforcement in a the service consumer.

  In the service consumer, when access to data is granted:

  - It receives data flow information and the policies from the DMP of the service provider.
  - It deploys received policies in the PMP to be used for policy enforcement during data uses.
  - It updates received data flow information in the PIP to be requested by the PDP at policy enforcement.

  In turn, the DMP component manages the information shared in the distributed architecture through the distributed database. To do so, triggered by the PDP:

  - It requests information in the distributed database.
  - It updates shared information in the distributed database.

- Distributed database: It stores information shared along the distributed architecture that may affect the enforcement of policies.

With respect to the privacy-preserving approach, it manages the enforcement of policies during and after data usage in the service consumer side, but also, before usage at the service provider side. Thus, we consider that this is a suitable approach to control data through its life cycle.

Furthermore, if access to data is granted, it manages some new issues that were not previously supported related to the enforcement of policies in a distributed architecture. These issues are data flow tracking and coordination of policies. We think that this results in a richer approach where data usage can be controlled considering usage restrictions that were not able to be addressed in previous approaches.

2.3.2.4 Label-based Distributed Usage Control

Label-based Usage CONtrol (LUCON) (Schuette & Brost, 2018) is an open source DUC solution that proposes an approach to control how data is shared and processed between services based on message labeling.

It is worth mentioning that LUCON is limited to a centralized architecture where all the services are deployed in the same computer. However, it is being developed with the goal of controlling how data is used by services deployed in a distributed architecture. Therefore, we consider of interest to describe this approach.

LUCON relies on a routing engine that defines how data is shared between services based on routes. On this basis, LUCON decides if a service configured on the route can use or share the data by means of the evaluation of what are called "flow rules" based on the labels attached to the data and service descriptions. Flow rules and service descriptions are described below.

- Service descriptions: These declare information about the services among which the following stand out:
    - Identifier: A unique identifier of a service to which any flow rule may refer to.
    - Capabilities: Description of actions that a service can execute. Examples are logging or data deletion.
    - Labels removal: List of labels that will be removed from outgoing data of the service.
    - Labels creation: List of labels that will be added to outgoing messages of the service.
- Flow rules: define which services can use the data based on their identifiers, their capabilities and the labels received attached to the data. Figure 2.10 presents an example of a flow rule using a LUCON propietary language. It defines that a service

with an id 'consumerid' is allowed to use the data labeled as 'purposeOfUse' only if it can delete it after 30 days. Otherwise, the data will be dropped.

```
flow_rule {
  id deleteAfterOneMonth              // Rule id
  description "Deletes all data after 30 days"
  when consumerid                     // Service id
  receives {label(purposeOfUse)}      // Received message labels
  decide allow                        // Allow if requirements
fulfilled
  require delete_after_days(X), X<30  // Obligation
    otherwise drop                    // Alternative
}
```

Figure 2.10: LUCON flow rule example.

To control data usage, LUCON lacks a clearly defined architecture. As previously mentioned LUCON relies on a routing engine that defines how data is shared between services. Service descriptions and flow rules are centrally stored. On this basis, each time the data enters or leaves a service configured on the route, the flow rules are enforced. For the enforcement of flow rules, those flow rules defined for the service are obtained based on its identifier. In turn, these are filtered based on the labels received attached to the data. If there is a flow rule and the service has the capabilities required on the flow rule, data usage is granted. In addition, some labels are removed and added to the data as defined in the service description.

In short, today this approach is limited to a centralized architecture. However, as initially conceived, we consider that it will be applicable for DUC. It should be considered that flow rules lack of expressiveness. Thus, some usage restrictions can not be expressed. Therefore, we consider LUCON only suitable for specific contexts.

2.3.2.5 Event-based Distributed Usage Control

MYDATA (Jung, Eitel, & Schwarz, 2014) is a commercial DUC solution that proposes an approach to control how data is used in a

distributed architecture, monitoring and intercepting system relevant events.

In this approach, policies are initially agreed and known by the two roles involved in data usage. As previously defined, these roles are the data provider and the data consumer. The data provider shares the data from the data sources. The data consumer receives and uses the data from the data provider. Based on agreed policies, this approach monitors and intercepts those system events that trigger their enforcement before, during and after data usage.

Based on the ABAC model architecture, MYDATA presents an architecture that is deployed in the data provider and the data consumer. Figure 2.11 represents the MYDATA architecture.

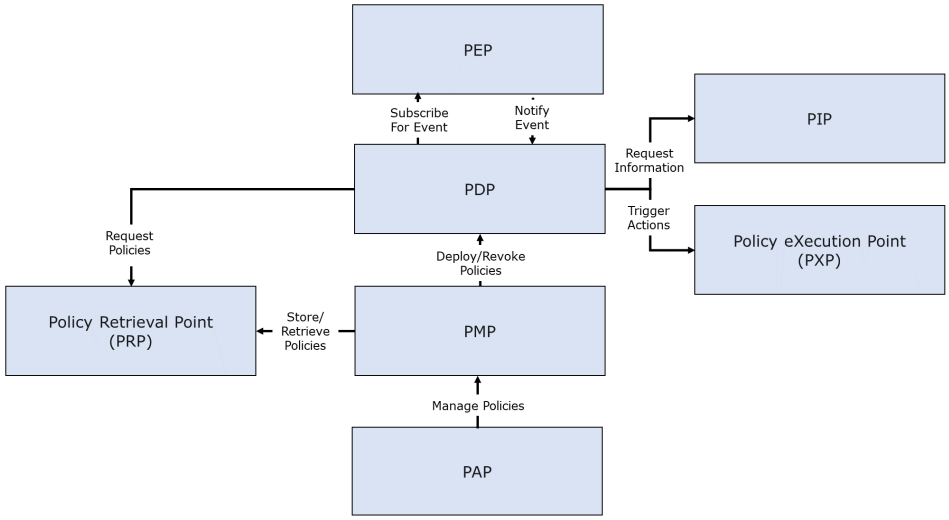

Figure 2.11: MYDATA architecture.

With respect to the ABAC model architecture, it includes three new components. These are: the PMP, the Policy Retrieval Point (PRP) and the Policy eXecution Point (PXP). All the components are describe below:

- PAP: It provides the interfaces to manage policies. Regarding

the component with the same name in the ABAC model architecture, it includes a set of new features (Hosseinzadeh., Eitel., & Jung., 2020). These are:

  - A policy editor to easily define policies following the IDSA UPL.
  - A policy transformation mechanism to translate IDSA UPL policies into MYDATA policies.
  - A mechanism to negotiate policies with other parties.

- PMP: It manages the policies. To do so, it has the following two functions:
  - It deploys and revokes the policies in the PDP.
  - It stores and retrieves the policies from the PRP.
- PRP: It is the policy storage place.
- PDP: It has two roles:
  - It enforces policies as in the ABAC model architecture. Nevertheless, the complexity of the policies in this context makes this process much more complex. To this end, while it evaluates conditions by requesting information to the PIP, it triggers actions to the PXP for the enforcement of duties.
  - It subscribes to different PEPs depending on the events that trigger the enforcement of policies.
- PEP: It monitors and intercepts events to trigger the enforcement of policies to the PDP. These are usage requests and events that trigger the enforcement of duties. Thus, decision continuity and obligation handling is supported. It should be noted that there are different PEPs depending on the type of event they monitor.
- PIP: It stores the information required by the PDP to evaluate policies as in the ABAC model.

- PXP: It executes actions related to duties.

With respect to the privacy-preserving and fully distributed usage control approaches, in this model policies are not propagated, but they are negotiated and known by the two parties involved in data usage. We consider this approach most suitable. This is because we think that requirements from the data consumer should be taking into account. Otherwise, data sharing may be jeopardize in some cases.

Based on agreed policies, it enforces policies before, during and after data usage as the fully decentralized approach. Thus, we consider it suitable to control how data is used through its life cycle in a distributed architecture.

However, the issues identified by the fully decentralized approach related to the enforcement of policies in distributed architectures are not addressed. These are data flow tracking and coordination of policies. It should therefore be noted that these aspects are not supported.

2.4 Assessment of Distributed Usage Control Solutions

As reviewed in Section 2.2, the different requirements to restrict the access to data led to the development of different strategies for implementing AC. Consequently, different AC models have been proposed (Penelova, 2021). From all existing AC models, the ABAC model is considered the basis to address UC (Sandhu & Park, 2003). As a result, based on the ABAC model, the reference framework for UC, the UCON model is presented (Aliaksandr et al., 2010). To address DUC, based on the UCON model, existing DUC solutions follow different approaches. These have been reviewed in Section 2.3.

This thesis aims to promote data sharing between organizations ensuring data sovereignty through the adoption of DUC solutions. In this regard, while existing solutions may follow different approaches to address DUC, there is not a reference framework to assess them. Therefore, it is difficult to identify to what extent existing DUC solutions ensure data sovereignty and the limitations that they need to overcome. We consider this assessment of utmost importance to be able to select the most suitable solution for an specific context as well as to face existing limitations in such a novel field as DUC.

On this basis, this thesis proposes a framework for the assessment of DUC solutions. In this regard, in the AC context, the comparison of AC models is a field where there has been a great deal of research. Considering the AC the basis for UC and DUC, we see of interest to review in this section the existing literature regarding frameworks for the assessment of AC models as it is a highly interesting starting point for developing a framework suitable for the assessment of DUC solutions.

2.4.1 Frameworks for the Assessment of Access Control Models

The frameworks proposed for the assessment of AC models are based on the comparative technique of the relative expressiveness analysis (Garrison & Lee, 2015b). This technique based on simulations compares the expressive power of the models. This refers to the ability of one model to replace another somehow.

Formal definitions of expressiveness have evolved between the frameworks proposed. We review below how the expressiveness has been defined throughout them. Based on this review, we also analyse the applicability that this comparative technique has to assess DUC solutions.

Initial researches formalize the concept of expressiveness in terms of highly specific properties within a complete set of models. For

example, the ability of the models to create new subjects and objects is compared (Ammann, Lipton, & Sandhu, 1992).

Other studies go a step further and address expressiveness by focusing on the implementation paradigm (Munawer & S, 1999; Nyanchama & Osborn, 1996; Osborn, Sandhu, & Munawer, 2000; Osborn et al., 2000; Sandhu, Coyne, Feinstein, & Youman, 1996; Sandhu & Munawer, 1998). That is, the ability of one model to replace another operationally.

Further on, the implementation paradigm was extended to include other considerations. This is the case of the theory (Tripunitara & Li, 2004, 2007) to compare the expressiveness of the models while preserving security properties (Harrison, Ruzzo, & Ullman, 1976; Li, Mitchell, & Winsborough, 2005).

Later, it was found that the techniques used to compare AC models ignored the scenario where they were to be deployed. Thus, the concept of parameterized expressiveness (Hinrichs et al., 2013) is introduced as the ability of a model to support a set of application-specific requirements. On that basis, a framework was presented that enables models to be compared based on the case of the use of the application. This framework is used in practical cases with different requirements (Garrison, Lee, & Hinrichs, 2014; Garrison, Qiao, & Lee, 2014).

Finally, the great variety of expressiveness formalizations used were synthesized (Garrison & Lee, 2015a). In addition, a set of minimum properties and additional properties is presented to provide a better understanding of the concept of expressiveness that best fits a particular case of use, the most appropriate comparison, and therefore the choice of the most suitable model.

In short, AC models following different strategies address specific requirements to restrict the access to the data. As a consequence, it can be deduced that comparing AC models not trivial. In this context, the relative expressive analysis arises as a comparative technique that is based on simulations. While the definition of expressiveness

in the initial framework is very limited, enables a limited analysis of specific properties of AC models and do not provide an adequate comparison of AC models, definitions from most current frameworks include various considerations such as the target scenario so that a more suitable comparison of AC models is provided. However, we consider the relative expressive analysis a high complexity technique.

On the contrary, in the DUC context, the requirements to restrict the usage of the data in a distributed architecture have evolved in the same direction between DUC approaches. As a result, as reviewed in Section 2.3.1, the specifications proposed to define policies that restrict the usage of the data in a distributed architecture also denoted as DUC policy languages have followed an incremental approach supporting new restrictions. The difference between DUC approaches reside in the architectures they present to control how data is used in a distributed architecture. Therefore, we think that there is no need to compare DUC solutions based on a complex comparative technique such as the relative expressive analysis. We consider of mayor interest to compare existing DUC solutions throughout the identification of the features their architectures must provide to comply with data sovereignty requirements in the context of data sharing and their assessment. To assess them, we see of interest to analyse the expressiveness of the policies each feature is able to deal with.

2.5 DUC Policy Quality

Policies are a critical aspect in AC, UC and DUC. They enable to express restrictions on the access to data in the AC context and the usage of the data in the UC and DUC context. It is important to consider that based on them, decisions are made on how data is accessed in AC and used in UC and DUC.

In this regard, the quality of the policies is a key aspect to consider as poorly defined policies can lead to security issues due to unauthorized accesses or uses of data, or they can cause performance

losses related to longer policy enforcement times. The former happens, for example, if a permission overrides a prohibition that applies for the same request. The latter, for example, if the same permission is defined multiple times. In this sense, the policy quality sets basic requirements that define if policies are well-defined so that the aforementioned security and performance issues are avoided. Due to the importance that avoiding these situations have on some contexts, the policy quality has been largely addressed. This is the case in the AC context.

The requirements that ensure the quality of AC policies have been already identified. These are consistency, minimality, relevance, completeness and correctness. Furthermore, several algorithms have been developed to analyse all of these requirements in AC policies. These algorithms have been classified based on the approach they followed. These types of approaches are formal methods, model checking methods, data mining methods, graph/tree-based modeling methods and mutation testing.

As reviewed in Section 2.2, restrictions on the usage of the data in the UC context are more complex with respect to the ones regarding the access to the data in the AC context. This makes the achievement of well-defined policies more difficult in UC with respect to AC. Furthermore, a review conducted on the UCON model (Aliaksandr et al., 2010) identifies policy quality as an important field of research that needs to be studied in UC. Nevertheless, it is a pending work.

As reviewed in Section 2.3.1, restrictions on the usage of the data in the UC context have evolved to address the needs in the DUC context. As a consequence, achieving the policy quality is more difficult in DUC with respect to UC. Furthermore, with the aim of boosting data sharing between organizations security and performance issues must be avoided. The former compromises data sovereignty and jeopardize the adoption of DUC solutions. The latter may render the adoption of DUC solutions not feasible for specific scenarios. Therefore, we consider the policy quality for DUC of utmost importance. However, it has not yet been addressed.

In this regard, this thesis studies the policy quality for DUC and proposes a policy analysis approach to analyse the policy quality for DUC policies. Due to the lack of research on policy quality in the DUC context and following a logical approach, we address the policy quality for DUC based on existing AC research literature. In this regard, we consider interesting to make the following analysis:

- Analysis of DUC policies: It is focused on the study of the new features that DUC policies present with respect to AC ones, that further affect the policy quality and should be considered for the analysis of the policy quality for DUC policies.
- Study of the AC policy quality requirements: It tries to identify the requirements defined for AC policies that should be met by good quality DUC policies and thus, should be analysed for DUC policies.
- Analysis of the AC policy quality analysis strategies: It makes an analysis of the approaches proposed for the analysis of the policy quality for AC policies that can be used for DUC policy quality analysis considering the distinct features introduced. It also identifies which approach is the most suitable one to address the analysis of the quality of DUC policies.

These analyses are described in the following subsections.

2.5.1 Analysis of DUC Policies

As mentioned above, restrictions on the usage of the data in the UC context are more complex compared to the ones related to the access to the data in the AC context. In turn, usage restrictions in the UC context have evolved to address new requirements in the DUC context.

As already mentioned in Section 2.3.1, DUC policy languages are specifications used to express policies that restrict the usage of the

data in the DUC context. The DUC policy languages proposed in the research literature have evolved in the same direction to support increasingly new requirements related to data usage. We consider that a detailed analysis of these DUC policy languages will enable us to identify the features that DUC policies present with respect to AC ones to met requirements related to data usage in the DUC context.

After a detailed analysis of the DUC policy languages, we conclude that policies in DUC introduce two main features with respect AC policies that further affect the policy quality. The features are detailed below.

- Heterogeneity of conditions: Due to the importance of context information for policy enforcement in distributed architectures, permissions and prohibitions on data usage are refined by much more numerous and diverse conditions than in AC.

- Extended control by supporting duties: To further control data usage, permissions are refined by supporting duties, which define the actions that must be executed under specific conditions.

On the basis of this study we consider that the analysis of the policy quality for DUC policies must be addressed based on AC approaches for policy analysis, considering these two new features.

2.5.2 AC Policy Quality Requirements

In the AC research literature, major efforts have been dedicated to the identification of the policy quality requirements. The first survey on policy analysis mechanisms (Aqib & Shaikh, 2014) identified consistency and completeness as the requirements that define the polity quality. Further research (Bertino, Jabal, Calo, Verma, & Williams, 2018) analyses the policy quality in more depth. As a result, they formalize policy quality according to the following requirements:

- Consistency: It means that policy enforcement does not result in a conflict. Two different type of conflicts may arise: modality conflicts and conflicts of rights. In turn, conflicts of rights are divided into conflicts of duty, conflicts of interest and inference conflicts.
    - Modality conflict: They arise if policy enforcement results in a permit and prohibit decision at the same time for a specific request.
    - Conflicts of rights: It refers to the assignment to a subject of more that one rights that it should not have at the same time.
        - Conflicts of duty: The separation of duty is a security principle that is based on the division of tasks and rights. To prevent errors and frauds, the consequent tasks must be carried out by different subjects with the corresponding rights. In this way, if a subject has the rights to perform more than one subtask, a conflict of duty appears.
        - Conflicts of interest: It appears when the two rights that lead to a conflict of interest are assigned to the same subject.
        - Inference conflicts: It emerges when a right that is prohibited can be inferred from other permissions assigned.
- Minimality: It refers to the fact that set of policies does not include redundant policies. Redundant policies refer not only to identical ones, but also policies that are unnecessary because they are already covered by more general others, e.g., two identical permissions for a subject and a role to which the subject belongs.
- Relevance: It ensures that the set of policies only applies for the requests that are given in a use case. In particular, they are implemented for rights that will be requested by subjects on resources.

- Completeness: It ensures that for each request there is a policy controlling it. That is, the rights that will be requested by subjects on resources are controlled.
- Correctness: It establishes that the set of policies is free of faults and implements its intended goals.

We think that consistency, minimality, relevance, completeness and correctness cover the requirements that should be met by good quality policies in DUC. Therefore, we conclude that all of them should be analysed for DUC policies.

2.5.3 AC Policy Analysis Approaches

The algorithms proposed in the AC research literature to achieve the policy quality follow different approaches for policy analysis. The first study of policy analysis mechanisms (Aqib & Shaikh, 2014) classifies them as formal methods, model checking, matrix-based approaches, mining techniques, mutation testing techniques and others. Further on, a more extensive and mature survey on policy analysis algorithms (Jabal et al., 2019) classifies them as formal methods, which include matrix-based methods, model checking methods, data mining methods, graph/tree-based modeling methods, and mutation testing. We consider this last classification more appropriate. This is because it takes into account a greater number of policy analysis algorithms. Furthermore, it covers the former classification and extends it. The classification proposed in this survey is described below.

- Formal methods: Involve mathematical concepts and techniques to specify a policy set. Within this approach the following methods are included: Matrix-based, Reasoning, Event Calculus and Argumentation.
- Model checking methods: These methods use a mathematical representation to define a finite state transition graph that

characterizes a policy set. This approach includes the following methods: Boolean Satisfiability Problem (SAT) and Solvers, Alloy and Binary Decision Diagrams.

- Data mining methods: They explore a massive policy set to detect patterns to guide for further policy analysis. Several methods exist within this approach: Association Rule Mining, Clustering, Data Classification and Role Mining.

- Graph/tree-based modeling methods: They consist in the representation of a set of policies in a policy graph or tree and the usage of graph operations for efficient queries.

- Mutation testing methods: They generate mutant policies from the original policy set introducing faults. The percentage of mutant policies whose enforcement output is different from the original policies is calculated.

As mentioned before, to promote data sharing through DUC, we state that the quality of DUC policies must be ensured. Therefore, we only establish as suitable for DUC those approaches that, considering the features introduced in DUC policies with respect to AC policies, are able to completely analyse the policy quality for DUC policies. That is, those that are able to detect any issue related to each of the policy quality requirements.

In this regard, formal and model checking methods present a high cost in terms of development effort. Policies in DUC become too complex due to the features introduced. This makes very difficult to develop an algorithm capable of analysing all the policy quality requirements for DUC policies following these approaches. Therefore, we do not consider these methods suitable for DUC.

When it comes to data mining methods, they detect patterns in a set of policies explored. Based on these patterns, they analyse the policy quality afterwards. If a pattern that affects the policy quality is not detected, policy quality will not be completely analysed. The features introduced in DUC policies make it very difficult to detect

all the patterns needed to analyse the quality of DUC policies. This makes us to consider data mining methods not suitable for DUC.

The mutation testing methods only analyse correctness policy quality requirements. As all the requirements that set the policy quality for DUC must be achieved, we do not think that this approach is suitable for DUC.

With regard to the graph/tree-based modelling methods, they enable the analysis of all the policy quality requirements representing policies as tree or graph structures. Therefore, we consider the graph/tree-based modelling approach valid for the analysis of the policy quality for DUC. Moreover, as already reviewed in Section 2.5.1, policies are defined as structures composed of different interrelated components. Thus, it is not difficult to represent policies as tree or graph structures. In turn, based on these structures, searches can be quickly performed detecting all the policy quality issues. Therefore, we believe that this approach is suitable for the analysis of the policy quality for DUC.

In summary, we consider the graph/tree-based modelling approach is the most suitable method to analyse the quality for DUC policies. Therefore, to define our approach for the analysis of the policy quality for DUC policies, we consider of interest to review below specific AC graph/tree-based modeling policy analysis algorithms.

2.5.3.1 Graph/tree-based Policy Analysis Algorithms

In the AC research literature, graph/tree-based algorithms have been developed for different AC models. For each AC model, this section reviews the policy quality requirements analysed and the context in which they do so.

In the RBAC model, the graph/tree based policy analysis algorithm proposed analyses the consistency policy quality requirement (Huang, Sun, Wang, & Si, 2009). In particular, it analyses conflicts of duties for subjects based on the rules defined for the roles

to which they belong.

For the Purpose AC model, the consistency policy quality requirement is analysed for rules defined for the rights of the subjects over objects that are refined through usage purposes, which are hierarchically classified (Sun, Wang, Tao, Zhang, & Yang, 2011). Consistency is addressed in a novel way. The concept of inference is introduced. It is defined that if a usage purpose is already permitted for a right of a object on a resource, dependent usage purposes can not be permitted as they are prohibited by the inference of the permission.

For ABAC, only time-based environment attributes are considered. Also, only the consistency policy quality requirement is analysed. The first algorithm proposed (Aqib & Shaikh, 2014) ensures that there are not contradictory decisions for a subject to perform a right on an object in overlapping time intervals. Further algorithm (Deng & Zhang, 2017) introduces the transmission relationship of access authority. It defines how permission and prohibition rights for subjects are inherited by dependent resources. In this regard, it not only analyses contradictory decisions for a subject to perform a right on an object in overlapping time intervals, but also, it analyses that for a subject there are not conflicts between inherited permissions or prohibitions for a right on an object and those explicitly defined.

As a conclusion, from the AC research literature, the algorithms presented for policy analysis following the graph/tree-based modeling approach are oriented towards the analysis of the consistency policy quality requirement. Furthermore, each algorithm only analyses one type of condition. As an example, for the ABAC model only time-based conditions are analysed. However, in DUC policies more type of conditions could be defined and should be considered for their analysis. For example, location-based conditions. Moreover, the conditions analysed in each algorithm are always expressed homogeneously. As an example, for the ABAC model, time is always defined through time ranges. In DUC policies, a time-based condition could be also defined through an event. It should be considered in

the analysis. In addition, the concept of condition is over-simplified. Depending on whether rules are of permission or prohibition, their refinement with conditions has inferences for dependent but non-overlapping conditions. That is, a time-based condition refining a permission inferences a prohibition on the rest of the time. However, this issue is not considered for policy analysis. This should be done. Otherwise, consistency will not be fully ensured. Finally, for the ABAC model, the idea that consistency should be ensured not only for the policies defined for a resource, but also, for dependent resources is introduced. However, for the ABAC model, this issue was addressed presenting the above-mentioned limitations. This is also important for DUC policies. Nevertheless, it has to be correctly addressed so that consisenty between the policies defined for dependent resources is completely ensured.

2.6 Conclusions

This thesis has the objective of promoting data sharing between organizations to enable the benefits it presents. In this regard, data sovereignty is considered crucial. That is, the self-determination of organizations about the usage of their data must be granted. To grant data sovereignty in data sharing scenarios, we consider that DUC is the most suitable approach. Therefore, the contributions of this thesis are focused on DUC.

This chapter reviews existing literature related to the contributions of this thesis. The technological context of DUC is presented. Moreover, the approaches proposed by existing solutions to address DUC are analysed. Furthermore, research related to the assessment of DUC solutions and the policy quality for DUC is examined.

As a first step towards DUC it is important to consider AC. It prevents the access to the data in an authorized manner. To do so, it controls restrictions on the access to the data. Nevertheless, if data is accessed, AC does not control how it is used afterwards. UC goes

a step further and regulates how data can be used through its life cycle, from its access to its several uses. In this regard, it extends AC strategies to control future restrictions on data usage. From this general definition of UC, this thesis differentiates between UC and DUC. UC addresses those scenarios where data is retained and used in a IT system of an organization. We define this scenario as a centralized architecture. DUC extends UC to address those scenarios where data from an organization is shared with another organization and used in its IT system. We define this scenario as a distributed architecture.

Motivated by different requirements to restrict the access to data, different strategies have been followed to address AC and different AC models have been proposed. The most widespread are RBAC, MAC, DAC and ABAC. The ABAC model restricts the access to the data based on authorizations that define whether subjects can access objects based on the requested rights and the attributes of the subjects, the objects and the environment. To control access to the data, it proposes an architecture based on different components that provide the interfaces to deploy and revoke authorizations, intercept access requests, evaluate authorizations and store all the attributes from subject, objects and the environment required for the the evaluation of authorizations. This model is established as the basis to address UC. We consider that this is for two reasons.

- Flexibility: The use of attributes enables to define complex restrictions to cover the requirements to restrict the usage of the data in the UC context.

- Context-Awareness: The attributes allows to control data in changing environments such as the data life cycle.

Based on the ABAC model, the UCON model is proposed to address UC. With respect to the ABAC model, the UCON model introduces the following contributions:

- To restrict data usage, it extends the concept of authorizations of the ABAC model and introduces conditions and obligations. Conditions are covered in the ABAC model within authorizations. However, obligations are a new concept introduced by the UCON model that is divided into subject and system obligations.

- To control data through its several uses, it introduces the concepts of decision continuity and obligation handling.

- To control data usage, it proposes an architecture based on the ABAC model architecture that introduces a new component to manage decision continuity and obligation handling and extends the scope of the evaluation of authorizations to also evaluate conditions and enforce obligations.

Based on the UCON model, existing DUC solutions follow different approaches to address DUC. To analyse them we consider that there are two aspects that should be taking into account. These are the specifications used in DUC approaches to express the policies that restrict on the usage of the data in a distributed architecture also denoted as DUC policies languages and the components of the architecture proposed to control data usage in a distributed architecture also denoted as DUC architectures.

The capabilities and complexity of the policy languages in DUC have evolved to address new requirements related to data usage in the DUC context. In particular, the concepts used in DUC policy languages have been refined and their scope increased from PPL and OSL, to ODRL and the IDSA UPL. We consider that a DUC policy language should consider two aspects to reflect restrictions related to data usage. These are permissions and prohibitions that define the conditions under which a usage request is permitted or prohibited and duties that set actions that must be performed as a result of a data permission. On this basis, we consider the reference DUC policy language the one that enables the definition of much diverse conditions related to permissions and prohibitions and actions related

to duties. Based on these considerations, we believe that the IDSA UPL is the reference policy language to date for DUC.

To control data usage in a distributed architecture, different approaches have been proposed. Examples are the observation-based, the privacy-preserving, the fully decentralized, the label-based and the event-based. As a result, DUC approaches present different architectures composed of multiple components.

Since existing DUC solutions may follow different approaches to address DUC, it is difficult to identify to what extent they ensure data sovereignty and the limitations that they need to address. Furthermore, there is not a reference framework to assess DUC solutions. We consider this assessment of utmost importance to be able to select the most suitable solution to address DUC in a particular context as well as to address existing limitations in such a novel field as DUC.

In this regard, in the AC context, which constitutes the basis for UC and DUC, the comparison of AC models is a field where there has been a great deal of research. AC models following different strategies address specific restrictions to control access to the data. Thus, comparing them is not trivial. To assess AC models different framework has been proposed. All of them are based on the relative expressive analysis, a comparative technique that analyses the ability of one model to replace another based on simulations. Between these frameworks, the comparison of AC models have evolved. While the initial framework is very limited, enable a limited analysis of specific properties of AC models and do not provide an adequate comparison of AC models, definitions from most current frameworks include various considerations such as the target scenario so that a more suitable comparison of AC models is provided. However, we consider this technique of high complexity.

In the DUC context, as already mentioned, the restrictions to control the usage of the data have evolved in the same direction between DUC approaches. This is evidenced by the fact that policy languages in DUC have followed an incremental approach including new restrictions. Therefore, we think that there is no need to com-

pare DUC solutions based on a complex comparative technique. The difference between DUC approaches reside in the architectures they present to control how data is used in a distributed architecture. Therefore, we consider of interest to compare existing DUC solutions throughout the identification of the features that their architectures must provide to met data sovereignty requirements in the context of data sharing and their assessment. To assess them, we consider of interest to analyse the expressiveness of the policies each feature is able to deal with. Following this approach, the Framework for the Identification and Assessment of Main Features of Distributed Usage Control Solutions, presented in Chapter 3, is built.

One of the key aspects in AC, UC and DUC is the policy quality. Policies define restrictions on the access to data in the AC context and the usage of the data in the UC and DUC context. Enforcing them, decisions are made on how data is accessed in AC and used in UC and DUC. Poorly defined policies lead to security vulnerabilities due to unauthorized accesses or uses that should not be allowed. In addition, low quality policies cause longer policy enforcement times than necessary resulting in performance losses. In this regard, policy quality sets basic requirements that define if policies are well-defined so that their enforcement does not lead to unauthorized data uses and wasteful policy enforcement times.

To promote data sharing between organizations through the adoption of DUC solutions, we consider that these undesired situations related to poorly defined DUC policies must be avoided. The former compromise data sovereignty and increase the reluctance to share data. Thus, security issues jeopardizes the adoption of DUC. The latter may render the adoption of DUC not feasible for scenarios with specific performance requirements. Therefore, we consider the policy quality for DUC of utmost importance. However, it has not yet been addressed.

In the AC context, the policy quality has been largely addressed. The requirements that set the quality of AC policies have already been defined. These are consistency, minimality, relevance, com-

pleteness and correctness. Furthermore, several algorithms have been developed to analyse them requirements for AC policies. These algorithms have been classified based on the strategies they followed. These approaches are formal methods, model checking methods, data mining methods, graph/tree-based modeling methods and mutation testing.

Following a logical approach, we consider interesting to address the policy quality for DUC based on the AC defined methods. To do so, we have analysed the following aspects:

- The two features introduced by DUC policies regarding the ones in AC context that further affect the policy quality. These are context-aware control and extended control by supporting duties. These features should be considered for the analysis of the policy quality for DUC policies.

- Consistency, minimality, relevance, completeness and correctness have been defined as the requirements that set the quality of AC policies. We see these requirements suitable to set the quality of DUC policies. These requirements should be analysed for DUC policies.

- The approaches that have been proposed for the analysis of the policy quality for AC policies are formal methods, model checking methods, data mining methods graph/tree-based methods and mutation testing. We only consider suitable for DUC those approaches from AC that are able to completely analyse the policy quality for DUC policies considering the features introduced with respect to AC policies. Otherwise, security and performance issues arise, which make the adoption of DUC not feasible. We believe that only the graph/tree-based modelling approach is suitable for DUC.

Considering these aspects, the Context-Aware Policy Analysis Algorithm for Distributed Usage Control is presented in Chapter 4.

Chapter 3

Framework for the Identification and Assessment of Main Features of Distributed Usage Control Solutions

3.1 Introduction

To encourage data sharing between organizations, which is the objective of this thesis, data sovereignty is fundamental. This refers to the self-determination of organizations regarding the usage of their data. To grant it in the context of data sharing, we think that two requirements should be considered. First, policies defining restrictions on the usage of the data must be implemented considering the needs of the two organizations involved. Second, implemented policies must be enforced from its access to its multiple uses once it has been shared. To met these policy implementation and enforcement requirements, we consider DUC as the most suitable approach.

As reviewed in Section 2.3, existing DUC solutions have followed different approaches to address DUC supporting different features. We think that it is interesting to identify the features that must be supported by DUC solutions to met the aforementioned policy implementation and enforcement requirements. This will set the path to ensure data sovereignty through DUC. Furthermore, as indicated in Section 2.4 there is not a reference framework to assess DUC solutions following different approaches. As a result, it is difficult to identify the strengths and weaknesses that DUC solutions present to ensure data sovereignty. We consider this assessment of of great interest. Thus, the most suitable solution for a particular context can be selected. Moreover, the limitations of existing solutions addressed.

As reviewed in Section 2.4.1, in the AC context, which constitutes the basis for UC and DUC, the comparison of AC models is a field of research where there has been a great deal of work. Different AC models arise motivated by different requirements to restrict the access to the data. It can then be deduced that comparing them is not trivial. In this regard, the frameworks proposed for the assessment of AC models are based on the expressive power of the models, a technique to compare the ability of one model to replace another.

On the contrary, in the DUC context, the requirements to restrict the usage of the data have evolved in the same direction between DUC approaches. As reviewed in Section 2.3.1, the specifications in DUC approaches to express policies that restrict the usage of the data also denoted as DUC policy languages have followed an incremental approach supporting new restrictions. As analysed in Section 2.3.2, what changes among DUC approaches is the features that the architectures proposed to control how data is used. Therefore, we believe that there is no need to compare DUC solutions based on a comparative technique such as the relative expressive analysis. Instead, we consider more interesting to compare DUC solutions identifying the features related to the implementation and enforcement of policies that their architectures must provide to grant data sovereignty and their assessment. To assess them, we consider interesting to analyse the expressiveness of the policies each feature is able to deal with.

This chapter describes a framework for the identification and assessment of the main features of DUC solutions. We see that a detailed analysis of existing research related to AC, UC and DUC is a good approach to identify the features that should be supported by DUC solutions to met policy implementation and enforcement requirements related to data sovereignty in the context of data sharing. This analysis is presented and the resulting features identified. Thus, the path to ensure data sovereignty through DUC is set. Furthermore, considering the features identified, a framework for the assessment of DUC solutions is presented. As a result, it is possible to identify to what extent DUC solutions ensure data sovereignty. Thus, the most suitable DUC solution for a particular context can be selected and the limitations of existing DUC solutions can be identified.

3.2 Identification of Main Features for DUC Solutions

DUC controls how an IT system of an organizations uses the data provided by another organization. This involves two roles. These are the data provider and the data consumer.

- Data provider: It is the organization that shares the data from its data sources with the data consumer.
- Data consumer: It is the organization that receives and uses the data from the data provider in the IT system.

To address DUC, we state that a client-side and server-side DUC solution must be deployed. While the data provider is responsible for the client-side, the data consumer is responsible for the server-side. Based on this consideration, Figure 3.1 represents a general architecture of a system where DUC is implemented.

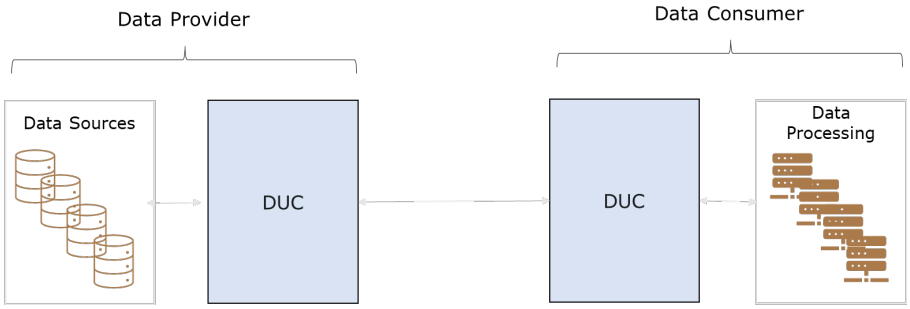

Figure 3.1: General architecture of DUC.

As initially remarked, to ensure data sovereignty in the context of data sharing we consider that DUC must enable the implementation of policies defining restrictions on the usage of the data considering the needs of the data provider and the data consumer and the enforcement of these policies from its access to its several uses. Considering DUC approaches, how we think that DUC should perform this is described below:

- Policy implementation: Data providers define policies on how their data can be used. These are denoted as offered policies. Data consumers define policies too on how they would like to use the data. These are denoted as requested policies. From offered and requested policies, agreed policies are established and implemented. This process is represented in Figure 3.2.

- Policy enforcement: Agreed policies are enforced on both sides to control a usage request from the data consumer to the data provider and then on the data consumer to control usage requests of the received data. This process is represented in Figure 3.3.

From the existing research related to AC, UC and DUC, in this section we identify and describe the features that DUC solutions must support to address the policy implementation and enforcement requirements.

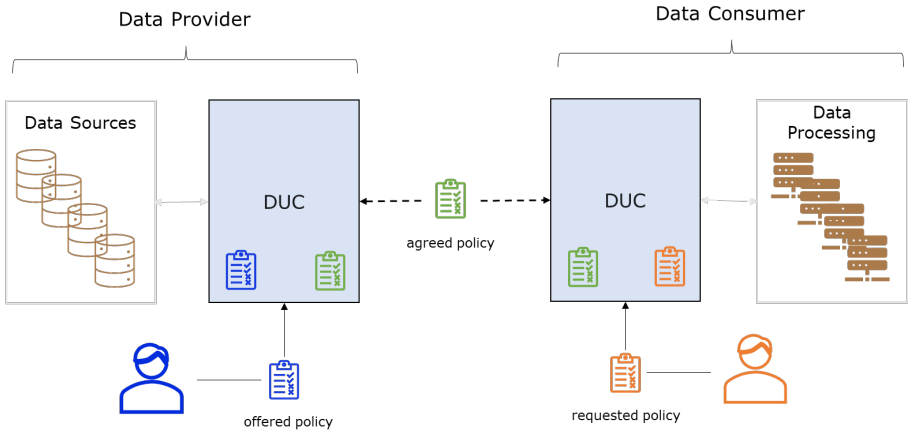

Figure 3.2: DUC. Policy implementation.

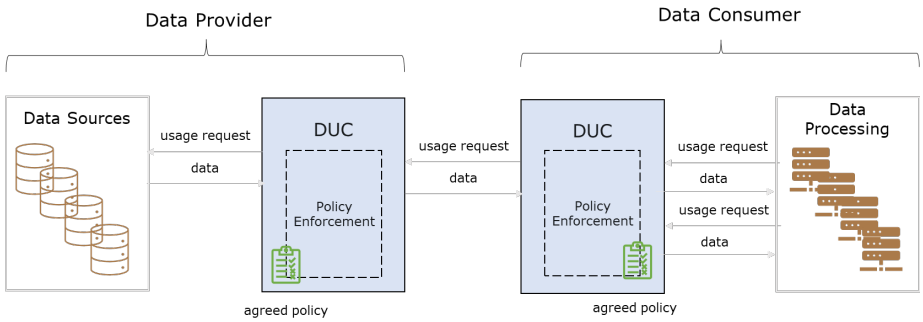

Figure 3.3: DUC. Policy enforcement.

3.2.1 Policy Implementation

Considering the most relevant research related to the implementation of policies from AC, UC and DUC, we consider that DUC solutions must support the following features for the implementation of policies that restrict the usage of the data: policy specification, policy quality, policy negotiation and policy transformation. Figure 3.4 represents how these features, supported in a DUC solution in the data provider premises and the data consumer scenario, interact with each other. They are described below.

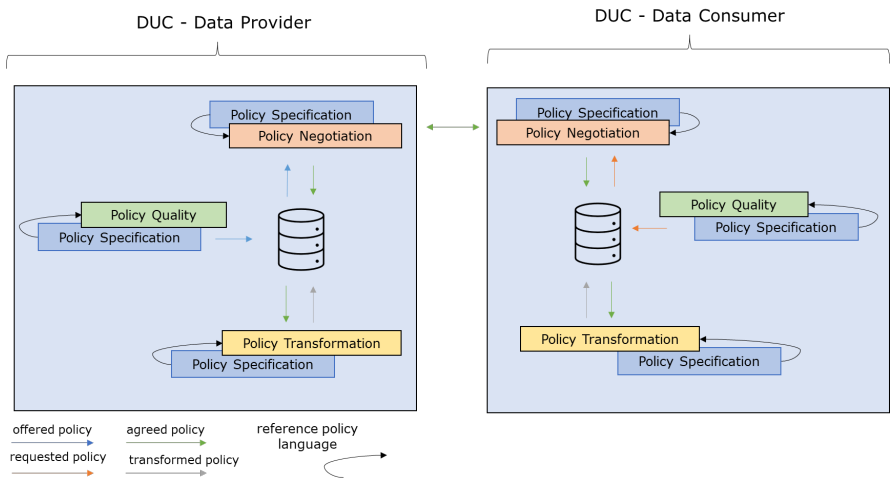

Figure 3.4: DUC. Policy implementation features.

3.2.1.1 Policy Specification

Data providers define offered policies on how their data can be used. Data consumers define requested policies too on how they would like to use the data. To enable the agreement of policies, interoperability between offered and requested policies must be ensured. In this regard, we consider suitable that DUC solutions should establish a reference policy specification for the definition of offered and requested policies.

For the specification of the policies it is essential to use a reference policy language that allows to express the different aspects related to the restrictions that may apply to the use of the data. We consider these aspects to be the permissions and prohibitions which define the context under which data use is permitted or prohibited and duties that define actions to be performed if data use is permitted. On this basis, we believe that the most expressive DUC policy language should be considered by DUC solutions as the reference for policy language. This reference DUC policy language will be the one that enables the definition of more diverse conditions related to permissions and prohibitions and actions from duties.

3.2.1.2 Policy Quality

Policies are a critical aspect in the DUC context. When policies are poorly defined, security issues may arise due to unauthorized data uses and performance losses can appear derived from longer policy enforcement times. The former happens, for example, if a permission overrides a prohibition that applies for the same request. The latter, for example, if the same permission is defined multiple times. For an adequate operation of DUC, these situations must be avoided. The former compromise data sovereignty and jeopardize the adoption of DUC. The latter may include additional delays on data sharing and render the adoption of DUC not valid for particular scenarios. In this sense, we consider that DUC solutions must ensure that policies, defined using the reference policy language, are defined with quality. The quality of the policies sets basic requirements that ensure that policies are well-defined so that unauthorized data uses and longer policy enforcement times than necessary are avoided.

It is interesting to mention that in the AC context, which constitutes the basis for UC and DUC, due to the importance that avoiding security and performance issues has, the policy quality has been largely addressed. Requirements that set the quality of AC policies have been defined. Furthermore, several policy analysis algorithms have been developed to analyse the quality of the policies with the final goal of achieving the quality requirements for AC policies. We see those policy analysis algorithms as the approach that DUC solutions must support to achieve the policy quality.

3.2.1.3 Policy Negotiation

Offered policies defined by data providers and requested policies from data consumers may differ. To enable data sharing, differences between offered and requested policies, if any, must be resolved and agreed policies established. With this aim, we define that DUC solutions must support the negotiation of policies that are defined making used of the reference policy language.

In the context of DUC, policy negotiation has been addressed for the first time in the event-based DUC approach. A theoretical approach to simulate human-like negotiations is proposed. However, this has not yet been implemented. The policy negotiation process should be included in DUC solutions.

3.2.1.4 Policy Transformation

To address DUC, data providers and data consumers can use different DUC solutions. Each DUC solution is able to enforce policies defined using an specific policy language. This may be the reference one or a proprietary one. To enable the enforcement of agreed policies, they must be implemented in the corresponding DUC solution by means of the utilization of the policy language that it understands. To do so, we define that DUC solutions must transform agreed policies defined using the reference policy language to agreed policies defined using their corresponding policy language.

The concept of policy transformation was introduced by the observably-based DUC approach to provide interoperability between OSL and Digital Rights Management (DRM) mechanisms. What is more interesting is the policy translator that the event-based DUC approach includes in the policy implementation process to generate MYDATA policies from policies defined using the IDSA UPL. We see this policy translator as the basis to ensure interoperability between DUC solutions understanding specific proprietary policy languages.

3.2.2 Policy Enforcement

Through an analysis of the research in policy enforcement architectures for AC, UC and DUC, we define that DUC solutions must support the following features for the enforcement of policies from the data access to its several uses: decision continuity, permission/prohibition enforcement, duty enforcement and duty handling. They are described below.

3.2.2.1 Decision Continuity

To ensure data sovereignty in the context of data sharing, DUC has to control what must happen to data from its access to the moment when it has been shared and is used multiple times. To this end, we state that DUC solutions must support decision continuity. The UCON model introduces the concept of decision continuity for a centralized architecture. However, to address it in a distributed architecture we consider that DUC solutions must follow another approach. This is described below.

Both the data provider and the data consumer monitor and intercept usage requests. In DUC, agreed policies are enforced twice before the usage session. In the data provider and the data consumer. If the data consumer intercepts a request for the usage of the data stored in the data provider, agreed policies are first enforced before the usage session in the data consumer. How policies should be enforced is described in the following sections. If policies are complied, the usage request is forwarded to the data provider. Otherwise, data usage is denied. In the data provider agreed policies are enforced again before the usage session. If satisfied, the data is shared with the data consumer and the usage session initiated. Otherwise, data usage is denied. During the usage session, the data consumer monitors and intercepts further usage requests on the received data. If intercepted, agreed policies are enforced again during data usage. The session will only be ended if data is deleted in the data consumer.

Figure 3.5 represents the decision continuity state diagram that we propose for DUC. Solid arrows refer to transitions triggered by external agents. These are the usage request and data deletion. Dashed arrows refer to transitions triggered by the DUC system. This is the case of the results of policy enforcement.

To support decision continuity, we consider that DUC solutions should monitor and intercept usage requests to trigger policy enforcement. Following the architectures proposed from DUC approaches this should be done by the PEP component.

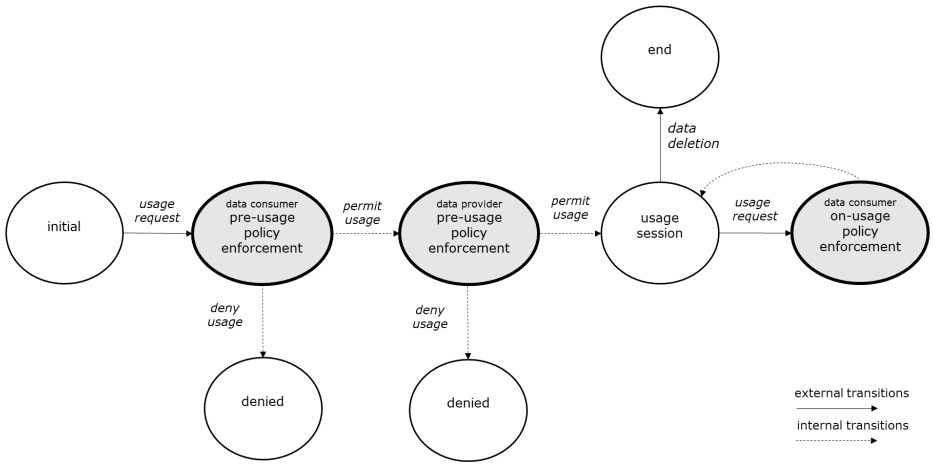

Figure 3.5: Decision continuity state diagram for DUC.

3.2.2.2 Permission/Prohibition Enforcement

To control data usage requests DUC must trigger the enforcement of policies. As previously commented, we consider that policies in DUC must reflect two different aspects related to the restrictions on the usage of the data. These are the permissions and prohibitions which define the context to permit or prohibit data usage and the duties that must be performed if data usage is permitted. In this way, to control usage requests, policies must be enforced. For the enforcement of policies, we define that DUC solutions must first enforce permissions and prohibitions.

To enforce permissions and prohibitions, DUC solutions must evaluate the conditions that refine when they apply. To this end, DUC solutions should gather all the information required for the evaluation of the conditions and evaluate the conditions themselves. To gather all the information required for the evaluation of conditions, DUC approaches proposed a component within their architecture called the PIP. Furthermore, the fully decentralized DUC approach includes within its architecture a component called DMP

to gather information shared in a distributed architecture. This includes data flow tracking information and global information such as the number of times data has been used. Therefore, we consider that information should be gathered by DUC solutions through a PIP and a DMP. To evaluate the conditions, all DUC approaches propose the same component within their architecture that we consider mandatory for the evaluation of conditions. This is the PDP.

3.2.2.3 Duty Enforcement

To further control usage requests through the enforcement of policies, apart from permissions and prohibitions, we state that DUC solutions must enforce duties.

To this end, two aspects related to the duties should be considered. These are the action that must be performed and the conditions that must be followed to do so. Therefore, we define that DUC solutions must execute actions related duties following the conditions that refine them. From the study of DUC architectures, this should be performed by a module called the PXP. In the privacy-preserving DUC approach, duties are performed by the obligation engine. However, the scope is the same with respect to the PXP proposed by other DUC approaches.

3.2.2.4 Duty Handling

Duties must be enforced if the usage request is permitted as a result of the enforcement of permissions and prohibitions. Nevertheless, when they must be enforced, additionally depends on the occurrence of events. These events may be the data usage permit itself as a result of policy enforcement, the expiration of a period of time, etc. As a result, duties are actions that must be performed if data usage is permitted and triggered by an event. Therefore, to enforce duties, DUC solutions must handle duties.

To this end, we consider that the DUC architecture should include a module responsible for handling events that trigger the enforcement of duties. In the privacy-preserving DUC approach, the event handler is introduced to this end. In the event-based DUC approach, events are managed by different PEPs.

3.3 Framework for the Assessment of DUC Solutions

In this section we present a framework for the assessment of DUC solutions. It is based on the features identified and the technology readiness level.

3.3.1 Policy Implementation

In Section 3.2.1 we define that DUC solutions must support the following features for policy implementation: policy specification, policy quality, policy negotiation and policy transformation. In what follows, we propose a formal approach to assess each of them.

3.3.1.1 Policy Specification

Based on the offered policies that data providers set and the requested policies that data consumers define, agreed policies must be established. To build agreed policies, interoperability between policies must be ensured. To this end, we define that DUC solutions should establish a reference policy specification for the definition of policies using the most expressive DUC policy language.

The policy language for DUC has evolved as new requirements to restrict the usage of the data are considered. It should be mentioned that all the DUC policy languages take into account the aspects of

permissions, prohibitions and duties that we have previously establish as mandatory to reflect restrictions related to data usage. The main difference between DUC policy languages relies on the diversity of conditions related to permissions and prohibitions and actions from duties covered by each of these languages. In this regard, the most expressive DUC policy language for the definition of policies is the IDSA UPL.

Based on this consideration, we define that DUC solutions must use the IDSA UPL for the definition of policies. Thus, DUC solutions can be assessed based on whether they enable the definition of policies using the IDSA UPL.

3.3.1.2 Policy Quality

Poorly defined policies can lead to unauthorized uses and longer policy enforcement times. Thus, causing security and performance issues that jeopardize the adoption of DUC. To ensure an adequate operation of DUC, these situations must be avoided. With this aim, we state that DUC solutions must ensure the quality of the policies through a policy quality analysis algorithm.

DUC solutions can be assessed based on whether they address the policy quality. Nevertheless, DUC solutions can address the policy quality, but only ensure it for policies with a limited expressiveness. For example, for policies that only include permissions and prohibitions. Therefore, we propose to assess DUC solutions considering the expressiveness of the policies they managed. For the assessment of the expressiveness of the policies, we consider the most expressive and reference policy language for DUC, the IDSA UPL, as the basis.

In the IDSA UPL, policies are divided into permissions or prohibitions and duties. Permissions or prohibitions define the conditions under which data usage is permitted or prohibited. Duties complement a permission to set a duty action that must be executed if data usage is permitted. Therefore, DUC solutions can be analysed

based on whether they ensure the quality of policies that consider permissions or prohibitions and/or duties. This assessment should be refined considering the expressiveness of the conditions managed for permissions or prohibitions and duty actions set for duties. How this can be performed is described below.

Conditions

Figure 3.6 illustrates how the concept of condition is specified in the IDSA UPL. In particular, a condition (*ids:Constraint*) specifies a Boolean function that indicates whether a given requirement (*ids:LeftOperand*) executing an operation (*ids:Operator*) is within a range of values. Those values are expressed as raw data (*ids:RightOperand*) or a reference value (*ids:RightOperanceReference*). The *ids:Unit* indicates the unit of measurement of a *ids:RightOperand*. The *ids:PipEndpoint* points to the PIP needed to evaluate the *ids:Constraint*.

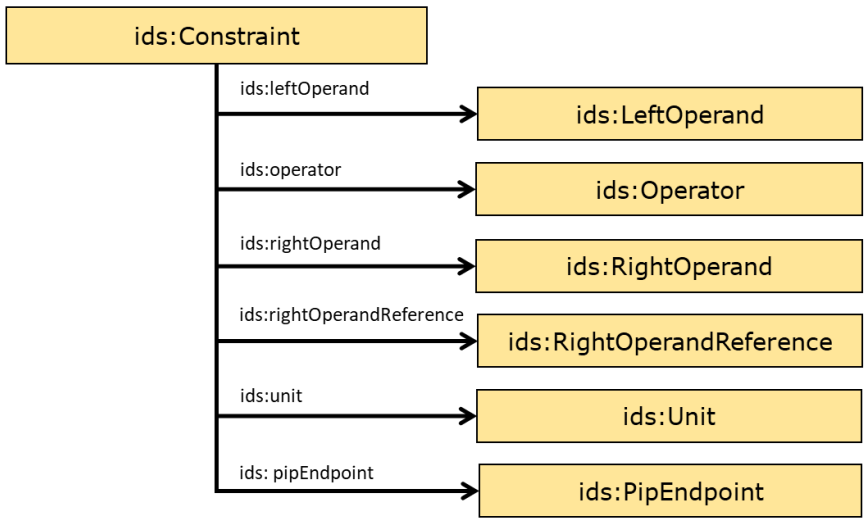

Figure 3.6: IDSA UPL condition.

Table 3.1 represents some examples of conditions. A user-based requirement (*idsc:USER*) can be defined based on a Uniform Resource Identifier (URI) through the *idsc:EQUALS* operator, but also, based on a role through the *idsc:HAS_MEMBERSHIP* operator. A time-based requirement (*idsc:POLICY_EVALUATION_TIME*) can be defined considering a time range through the *idsc:TEMPORAL_EQUALS* operator, but also based on a timestamp through the *idsc:BEFORE* operator. Thus, we can deduce that the combination of different leftoperands with different operators lead to the definition of multiple different conditions.

Table 3.1: IDSA UPL condition examples.

Requirement (*ids:leftoperand*)	**Operator (*ids:operator*)**	**Value (*ids:rightOperand/ ids:rightOperandReference*)**
idsc:USER	idsc:EQUALS idsc:HAS_MEMBERSHIP	Data consumer URI Data consumer role
idsc:POLICY_EVALUATION_TIME	idsc:TEMPORAL_EQUALS idsc:BEFORE	Time range Timestamp
idsc:ABSOLUTE_SPATIAL_POSITION	idsc:EQ idsc:IN	Geospatial position URI Latitude and longitude
idsc:SECURITY_LEVEL	idsc:SAME_AS idsc:isAnyOf	Security level URI List of security level URIs

Therefore, the expressiveness of the conditions is set and can be assessed based on the combination of leftoperands and operators that can be defined.

Duty actions

In the IDSA UPL, a duty action is defined by the action that must be executed and the conditions that must be followed to execute it. Table 3.2 provides some examples about duty actions. A data deletion (*idsc:DELETE*) can be defined to be performed after a period of time through the *idsc:DELAY* leftoperand and *idsc:DURATION_EQ* operator, but it can also be defined to be performed at a timestamp through the *idsc:DATE_TIME* leftoperand and *idsc:BEFORE* operator. Therefore, we can deduce that the combination of different actions and conditions lead to multiple different duty actions.

Table 3.2: IDSA duty actions examples.

Action (***ids:action***)	**Requirement** (***ids:leftoperand***)	**Operator** (***ids:operator***)	**Value** (***ids:rightOperand/ ids:rightOperandReference***)
idsc:DELETE	idsc:DATE_TIME idsc:DELAY	idsc:BEFORE idsc:DURATION_EQ	Timestamp Time duration
idsc:ANONYMIZE	idsc:MODIFICATION_METHOD	idsc:EQ	Method type: replace, delete, etc.

As a result, the expressiveness of the duties is set and can be assessed based on the combination of duties and conditions that can be defined. In turn, conditions can be assessed as previously described.

3.3.1.3 Policy Negotiation

Offered policies from data providers and requested policies from data consumers can differ. To enable data sharing, differences between offered and requested policies must be resolved and agreed policies established. Thus, we define that DUC solutions must include the negotiation of policies including a policy negotiation process.

DUC solutions can address policy negotiation but only be able to negotiate policies with a limited expressiveness. Thus, following the same approach as for policy quality, DUC solutions can be assessed based on whether they address policy negotiation and if so, based on the expressiveness of the policies considered for policy negotiation. To assess the expressiveness of the policies we establish the IDSA UPL as the reference.

3.3.1.4 Policy Translation

Data providers and data consumers can use different DUC solutions which enforce policies using different DUC policy languages. To enable the enforcement of policies, agreed policies must be imple-

mented in the corresponding DUC solution following the proprietary policy language. To do so, we state that DUC solutions must support the transformation of policies including a policy translator.

DUC solutions can address policy transformation. Nevertheless, the expressiveness of the policies transformed should be analysed. Therefore, DUC solutions can be assessed considering if policy transformation is supported. If so, the expressiveness of the policies should be studied. To assess the expressiveness of the policies we establish the IDSA UPL as the reference.

3.3.2 Policy Enforcement

In Section 3.2.2 we set out that DUC solutions must support these features for policy enforcement: decision continuity, permission/prohibition enforcement, duty enforcement, duty handling. This section presents an approach to assess DUC solutions based on them.

3.3.2.1 Decision Continuity

To ensure data sovereignty, data has to be controlled through its life cycle. To provide this control, we define that DUC solutions must support decision continuity in a distributed architecture as we propose in Figure 3.5. To do so, DUC solutions regardless of whether they are used in the data provider or data consumer must monitor and intercept usage requests to trigger policy enforcement.

Therefore, decision continuity can be assessed for DUC solutions considering if usage requests are monitored and intercepted.

3.3.2.2 Permission/Prohibition Enforcement

To control usage requests, policies must be enforced. We state that policies in DUC must include permissions and prohibitions as

well as duties. As a first step towards the enforcement of policies, we define that DUC solutions must enforce permissions and prohibitions. To this end, DUC solutions must evaluate the conditions that define when permissions and prohibitions apply. To do so, DUC solutions must gather all the information required for the evaluation of conditions as well as perform the evaluation itself.

Following the examples that we show in Table 3.1, the information required for the evaluation of each condition depends not only on its left operand, but also, the operator. A user-based requirement can be refined through a simple URI or by the role of the data consumer. Thus, while the evaluation of the former requires only the URI of the data consumer, which can be easily managed through a local storage, the role of the consumer may be managed through an external storage such as an Lightweight Directory Access Protocol (LDAP) server.

Furthermore, also depending on both the leftoperand and operator, conditions are evaluated differently. While the *ids:ABSOLUTE_SPATIAL_POSITION* requirement only needs a comparison between URIs when the *idsc:EQ* operator is used, if the *idsc:IN* operator is used, whether a position is within a latitude and longitude must be identified.

Therefore, we propose that DUC solutions can be assessed based on the combination of leftoperand and operators that can be evaluated for the enforcement of permission and prohibition rules. Based on the issues already identified, Table 3.3 presents an example for the assessment of the enforcement of permissions and prohibitions.

3.3.2.3 Duty Enforcement

To further control usage requests through the enforcement of policies, we state that DUC solutions must enforce duties. Duties are composed of the actions to be performed and the conditions to follow in this regard. Thus, DUC solutions must perform the actions under the conditions defined.

Table 3.3: Example of permission / prohibition enforcement assessment.

Requirement (***ids:leftoperand***)	**Operator** (***ids:operator***)	**Supported**	**Reason**
idsc:USER	idsc:EQUALS	Yes	—
	idsc:HAS_MEMBERSHIP	No	Unable to retrieve information from an external PIP (LDAP Server)
idsc:ABSOLUTE_SPATIAL_POSITION	idsc:EQ	Yes	—
	idsc:IN	No	The PDP can not evaluate a position based on latitude and longitude

The enforcement of duties depends on the execution of duty actions. These duty actions refer to actions performed following specific conditions. Following the example that we show in Table 3.2, the enforcement of a duty action does not only depends on the execution of the action itself such as *idsc:ANONYMIZE*, but also, the conditions that must be followed such as the method to anonymize the data.

As a result, we propose to assess DUC solutions in terms of the duty actions that can be performed considering these as a combination of an action to execute and the conditions to followed. Table 3.4 presents an example of this assessment.

Table 3.4: Example of duty enforcement assessment.

Action (***ids:action***)	**Requirement** (***ids:leftoperand***)	**Operator** (***ids:operator***)	**Supported**	**Reason**
idsc:DELETE	idsc:DATE_TIME	idsc:BEFORE	No	—
	idsc:DELAY	idsc:DURATION_EQ	Yes	Cannot deal with time durations
idsc:ANONYMIZE	idsc:MODIFICATION_METHOD	idsc:EQ	No	There is no access to data

3.3.2.4 Duty Handling

Duties are actions that must be performed if data usage is permitted and triggered by an event. Therefore, for the enforcement of duties we define that DUC solutions must handle the events that trigger the enforcement of duties. With this purpose, DUC solutions must monitor and intercept those events that trigger the enforcement of duties.

DUC solutions can be assessed considering the number of events that can manage to trigger the enforcement of duties. A clear example of this is the usage request itself, for example, to log the result of the policy enforcement. However, others can be considered such as the time for a periodic payment or data deletion.

3.4 Features to Assess in the Framework

To ensure data sovereignty in the context of data sharing, we have defined that DUC solutions must achieve a policy implementation and policy enforcement requirement. This section summarizes which are the features that DUC solutions must provide to met these requirements and how they can be assessed.

Figure 3.5 and Figure 3.6 respectively identifies the features that DUC solutions has to support regarding the implementation and enforcement of policies. In both figures, the following information is provided for each feature:

- What issue the feature addresses?
- How the feature addresses the issue?
- How the feature can be assessed?

Table 3.5: Features to assess in the framework related to policy implementation (PS = Policy Specification, PQ = Policy Quality ,PN = Policy Negotiation, PT = Policy Transformation).

Feature	**What issue it addresses?**	**How it addresses the issue?**	**How it can be assessed?**
PS	Provides interoperability between offered and requested policies to enable the agreement of policies	Uses a reference policy language	Identifying if policies are defined using the IDSA UPL
PQ	Avoids unauthorized data uses and longer policy enforcement times related to poorly defined policies to ensure an adequate operation of DUC	Implements of a policy analysis algorithm	Based on the IDSA UPL, analyzing the expressiveness of the policies that can be managed
PN	Resolves conflicts between offered and requested policies and establishes agreed policies to enable data sharing	Includes a policy negotiation process	Based on the IDSA UPL, analyzing the expressiveness of the policies that can be managed
PT	Implements agreed policies defined using a proprietary language to enable their enforcement	Includes a policy translator	Based on the IDSA UPL, analyzing the expressiveness of the policies that can be managed

Table 3.6: Features to assess in the framework related to policy enforcement (DC = Decision Continuity, P&PE = Permission & Prohibition Enforcement, DE = Duty Enforcement, DH = Duty Handling).

Feature	**What issue it addresses?**	**How it addresses the issue?**	**How it can be assessed?**
DC	Controls the usage of the data from its access to the moment when it has been shared and is used multiple times to ensure data sovereignty	Monitors and intercepts usage requests through a PEP to trigger policy enforcement	Considering if usage requests are monitored and intercepted
P&PE	Enforcement of policies related to permissions and prohibitions to control usage requests	Evaluates conditions through the PDP gathering the information required through the PIP and DMP	Based on the IDSA UPL, analyzing the conditions that can be evaluated
DE	Enforcement of policies related to duties to control usage requests	Performs the actions under the conditions defined through the PXP	Based on the IDSA UPL, analyzing the actions that can be executed
DH	Trigger the enforcement of duties to control usage requests	Monitors and intercepts events that trigger the enforcement of duties through an event handler	Considering the events that can be monitored and intercepted

3.5 Conclusions

This chapter presents a framework for the identification and assessment of the main features that DUC solutions must support to ensure data sovereignty in the context of data sharing.

To address DUC existing DUC solutions have followed different approaches supporting different features. To ensure data sovereignty in the context of data sharing we think that two requirements should be considered. First, policies defining restrictions on the usage of the data must be implemented considering the needs of the two parties involved. Second, implemented policies must be enforced each time data usage is requested from its access along the period of time when data has been shared and is used multiple times. Considering these requirements, we believe that it is important to identify the features that DUC solutions must support to ensure data sovereignty in the context of data sharing. This will set the path to ensure data sovereignty through DUC.

Furthermore, there is not a framework to assess these DUC solutions that follow different approaches. Therefore, it is difficult to identify the features and limitations that DUC solutions present to ensure data sovereignty. We consider this assessment of utmost importance for two reasons. This will enable to select the most suitable solution for a particular context. Furthermore, the limitations of DUC solutions can be identified to be addresses as future work.

To implement AC, which constitutes the basis for UC and DUC, motivated by different requirements to restrict the access to data, different strategies have been followed. As a result, different AC models have been proposed. In this context, the expressive power of the models, a technique to compare the ability of one model to replace another, is proposed as the approach to assess AC models.

In the DUC context, the requirements to restrict the usage of the data in a distributed architecture have evolved in the same direction. The differences that DUC approaches present is related to the ar-

chitecture they propose to control how data is used in a distributed architecture. On this basis, we think that there is no need to compare DUC solutions based on a complex comparative technique such as the relative expressive analysis. Instead, we propose to identify the features that DUC solutions must support for the implementation and enforcement of policies to grant data sovereignty in the context of data sharing. To assess the features identified, we see of interest to analyse the expressiveness of policies each feature can manage.

To identify these features, we assume two aspects of DUC. First, DUC involves a data provider and a data consumer. The data provider deploys the IT system that shares the data from the data sources with the data consumer. The data consumer deploys the IT system that receives and uses the data from the data provider. Second, a DUC solution must be deployed in the data provider and data consumer. Considering these aspects, we think that a detailed analysis of the most relevant research in AC, UC and DUC is a good approach to identify the features that DUC solutions must support regarding the implementation and enforcement of policies. The identification of these features is described below.

For policy implementation, we state that DUC solutions must support the following features: policy specification, policy quality, policy negotiation and policy transformation. Policy specification provides interoperability between offered and requested policies to enable the agreement of policies. It uses a reference DUC policy language for the definition of policies. Policy quality avoids unauthorized data uses and longer enforcement times related to poorly defined policies to ensure an adequate performance of DUC. It implements an algorithm for the analysis of the quality of the policies. Policy negotiation resolves conflicts between offered and requested policies and establishes agreed policies to enable data sharing. It includes a process for the negotiation of policies. Policy transformation implements agreed policies defined using the proprietary policy language of the DUC solution to enable their enforcement. It includes a translator of policies defined in the reference policy language to a proprietary policy language of the DUC solution.

For policy enforcement, we set that DUC solutions must support the following features for policy enforcement: decision continuity, permission/prohibition enforcement, duty enforcement, duty handling. Decision continuity controls the usage of the data through its life cycle to ensure data sovereignty. It introduces a PEP that monitors and intercepts usage requests to trigger policy enforcement. Permission/prohibition enforcement enforces permissions and prohibitions to control usage requests. It includes a PIP and DMP to gather the information required for the evaluation and a PDP to perform the evaluation itself. Duty enforcement enforces duties to control usage requests. It introduces a PXP to perform the actions defined in duties. Duty handling triggers the enforcement of duties to control usage requests. It includes an event handler that monitors and intercepts events that trigger the enforcement of duties.

Based on these features, we propose a framework for the assessment of DUC solutions. Using this framework, whether a DUC solution support these features can be identified. However, DUC solutions can support a feature such as policy quality, but only address it for policies with a limited expressiveness. In this regard, the framework enables the assessment of the expressiveness of the policies that a DUC solution manage for each feature. To assess the expressiveness of the policies, we consider the IDSA UPL as the reference. This is because of two facts. First, it includes all the aspects that we consider mandatory to reflect restrictions related to data usage. These aspects are the permissions and prohibition which define the conditions under which data use is permitted or prohibited and duties that define actions to be performed if data use is permitted. Second, is the DUC policy language that includes more diverse conditions related to permissions and prohibitions and actions from duties.

As a result, from the identification of the features that DUC solutions must support, the path to ensure data sovereignty in the context of data sharing through DUC is set. Furthermore, from the assessment framework, the features that DUC solutions support and their limitations can be identified. Thus, the most suitable DUC solution for a particular context can be selected. In turn, the limitations that

DUC solutions present can be identified to be addressed as future work.

Furthermore, the approach followed for the definition of the framework enables it to be updated according to the current technology readiness. New features can be added to or remove from the framework and the assessment can be adapted to new requirements to restrict the usage of the data in a distributed architecture.

Chapter 4

Context-Aware Policy Analysis Algorithm for Distributed Usage Control

4.1 Introduction

One critical aspect for the correct operation of DUC are the policies. They are used to establish the restrictions that govern the usage of the data in the distributed architecture. Based on them, DUC solutions make decisions on how data is shared from a data provider to a data consumer in the first place and afterwards how the data consumer uses the data. If policies are not accurate enough, they can lead to two undesired situations. First, DUC solutions may permit data usage unexpectedly. This happens, for example, if a permission overrides a prohibition that applies for the same request. Second, DUC solutions may take more time than necessary to make usage decisions. This occurs, for example, if the same permission is defined more than one time. In this regard, the policy quality sets basic requirements that define if policies are well-defined so that their enforcement does not lead to these security and performance problems.

To promote data sharing through the adoption of DUC solutions these situations must be avoided. The former compromise data sovereignty and jeopardize the adoption of DUC solutions. The latter may include additional delays on data sharing and render the adoption of DUC solutions not feasible for particular scenarios. Therefore, we consider the quality of the policies for DUC of utmost importance. However, it has not yet been addressed.

In the AC context, which constitutes the basis for UC and DUC the policy quality has been largely addressed. The requirements that set the quality of AC policies have been defined. Also, following different approaches, several algorithms have been developed to analyse these requirements for AC policies. Following a logical approach, this thesis addresses the analysis of the policy quality for DUC policies based on existing AC research literature. In this regard, some aspects should be considered. These are described below.

First, as described in Section 2.5.1, DUC policies introduce two features with respect to AC policies that further affect the policy quality. These are the heterogeneity of conditions and the extended control by supporting duties. These features should be considered for the analysis of the policy quality for DUC policies.

Second, as indicated in Section 2.5.2, consistency, minimality, relevance, completeness and correctness have been defined as the requirements that set the quality of AC policies. We consider these requirements also suitable to assess the quality of DUC policies. These requirements should be analysed for DUC policies.

Third, as reviewed in Section 2.5.3, the most relevant approaches that have been proposed for the analysis of the policy quality for AC policies are formal methods, model checking methods, data mining methods, graph/tree-based methods and mutation testing. Considering the features introduced in DUC policies with respect to AC policies that further affect the policy quality, the approaches that can be used to analyse the policy quality for DUC policies should be examined. Based on them, it is interesting to identify the most suitable one to address the analysis of the quality of DUC policies.

This chapter presents the approach that we define for the analysis of consistency and minimality policy quality requirements for DUC policies that support the heterogeneity of conditions based on the graph/tree-based modelling. In this regard, the aspects that we have considered to analyse the policy quality for DUC policies based on the AC research literature are described. Making use of the languages proposed for the expression of DUC policies reviewed in Section 2.3.1, a generic structure for DUC policies is presented. Regardless of the policy language used in a DUC solution, DUC policies can be mapped to this structure. Our approach analyses DUC policies based on this structure. Thus, it is valid for any DUC solution. According to this generic structure, several definitions and axioms that we assume for the analysis of DUC policies are introduced. Based on them, our approach for the efficient analysis of policies based on tree structures is presented. By efficient we mean that all consistency and minimality issues, which we denote as conflicts, can be detected in the shortest possible time. Also, the method to assess conflicts in the form of inconsistencies and redundancies is detailed. Finally, following the approach proposed, Context-Aware Policy Analysis Algorithm (CAPA) is presented. It is an algorithm that following the graph-tree based modelling approach analyses consistency and minimality policy quality requirements for DUC policies that support the heterogeneity of conditions also denoted as context-aware DUC policies.

4.2 Policy Quality Analysis for DUC Policies

As mentioned above, although we consider the policy quality extremely important for DUC, it is a field that has not yet been addressed. On the contrary, in the AC context, which constitutes the basis for UC and DUC, the policy quality has been largely addressed. Some requirements that set the quality for AC policies have been defined. Also, to analyse if a set of policies met these requirements, several algorithms have been developed following different approaches.

On this basis, this thesis addresses the analysis of the policy quality for DUC policies based the AC research literature. To this end, some aspects are considered. The new features introduced by DUC policies have been analysed. Also, the policy quality requirements for DUC policies have been identified based on the ones defined for AC policies. Furthermore, the policy analysis approaches for DUC policies have been considered based on the ones proposed for AC policies and considering the new features introduced in DUC policies.

This section presents the analysis of these aspects, including those considered in our approach for the analysis of the quality of DUC policies.

4.2.1 New Features of DUC Policies

As reviewed in Section 2.5.1, DUC policies introduce two main features with respect to AC ones that further affect the policy quality. These are heterogeneity of conditions and extended control by supporting duties. They are describe below:

- Heterogeneity of conditions: permissions and prohibitions are refined by much more numerous and diverse conditions than in AC. That is, while some conditions may apply to different domains (e.g., time or location), for the same domain, conditions can be heterogeneously described (e.g., time intervals or an event in time).
- Extended control by supporting duties: to further control data usage, permissions are refined by supporting duties, which define the actions that must be executed under specific conditions.

As discussed above, to promote data sharing it is important to avoid the security and performance issues related to poor quality DUC policies. However, we think that although performance is very important, security is the more critical aspect. This is because unauthorized data uses may put data sharing at risk as data sovereignty

is compromised. On the contrary, longer policy enforcement times can be assumed to some extent in certain situations or scenarios.

On this basis, while unnecessary policy enforcement can come from poorly defined permissions, prohibitions or duties, unauthorized data uses only appear from poorly defined permissions or prohibitions rather than duties. What is more, the heterogeneity of conditions makes the appearance of these situations more probable for DUC policies with respect to AC ones.

Therefore, although both features should be considered for the analysis of the policy quality for DUC policies, as we consider security more critical in DUC, our approach focuses on the analysis of DUC policies that support the heterogeneity of conditions. Based on this approach, the extended control by supporting duties can be addressed.

4.2.2 Policy Quality Requirements for DUC Policies

As stated in Section 2.5.2, we see the requirements that set the quality of AC policies, also suitable for granting the quality of the DUC policies. These are consistency, minimality, relevance, completeness and correctness.

To analyse these policy quality requirements, different sources of information are needed. To analyse consistency and minimality only DUC policies are considered. One step further, while transactions are also required apart from DUC policies to analyse relevance and completeness, the intended goals of the DUC policies must be known to analyse correctness.

As a first step towards the policy quality in DUC, our approach analyses the quality of DUC policies using a policy-based analysis. As a consequence, the scope of our approach is limited to the analysis of consistency and minimality policy quality requirements.

We think that these policy quality requirements are the most critical ones. While unauthorized data uses mainly came from inconsistencies between policies, longer policy enforcement times are caused above all by redundant policies. Therefore, we consider the scope of our approach enough to address the main issues that arise from poor quality DUC policies. Based on this approach, relevance, completeness can be addressed considering the analysis of the data usage transactions. In turn, correctness can be addressed through the analysis of the intended goals of the DUC policies.

4.2.3 Policy Analysis Approaches for DUC Policies

As analysed in Section 2.5.3, different approaches have been proposed for the analysis of the policy quality for AC policies. These are formal methods, model checking methods, data mining methods, graph/tree-based methods and mutation testing.

As mentioned before, to promote data sharing through DUC, we state that the quality of DUC policies must be ensured. Therefore, we only consider suitable for DUC those approaches from AC that are able to completely analyse the policy quality for DUC policies considering the features introduced with respect to AC policies. That is, those that are able to detect any issue related to each of the five policy quality requirements previously identified for DUC policies.

Formal and model checking methods present a high cost in terms of development effort. Policies in DUC become too complex due to the features introduced. This makes it is very difficult to develop an algorithm capable of analysing all the policy quality requirements for DUC policies following these approaches. Therefore, we do not consider these methods suitable for DUC.

Data mining methods detect patterns in a set of policies explored. Based on these patterns, they analyse the policy quality afterwards. If a pattern that affects the policy quality is not detected, policy

quality will not be completely analysed. The features introduced in DUC policies make it very difficult to detect all the patterns needed to analyse the quality of DUC policies. This makes us to consider data mining methods not suitable for DUC.

Mutation testing only analyses the correctness policy quality requirements. As all the requirements that set the policy quality for DUC must be achieved, we do not think that this approach is suitable for DUC. This is because this approach does not address consistency, minimality, completeness and relevance.

Graph/tree-based modelling enables the analysis of all the policy quality requirements representing policies as tree or graph structures. Therefore, we consider it valid for the analysis of the policy quality for DUC. Also, as already reviewed in Section 2.5.1, policies are defined as structures composed of different interrelated components. Thus, it is not difficult to represent policies as tree or graph structures. In turn, based on these structures, searches can be quickly performed for detecting all the policy quality issues. Therefore, we believe that this approach is suitable for the analysis of the policy quality for DUC.

That said, our approach analyses consistency and minimality policy quality requirements for DUC policies following the graph/tree-based modelling approach as we believe that this is the most suitable one for DUC.

4.3 Generic Structure for DUC Policies

In this section we define and formalize a generic structure for DUC policies considering the most significant features of the policy languages for DUC reviewed in Section 2.3.1. Our approach analyses DUC policies expressed by means of this structure. Regardless of the policy language used in a DUC solution, DUC policies can be mapped to this structure. Thus, it is valid for any DUC solution.

Formula 4.1 formalizes a policy p that is composed of a subset of rules $\{r_i\}, i = 1, ..., N$ where N is the number of rules.

$$p = \{r_i\}, i = 1, ..., N \tag{4.1}$$

A rule r_i describes the usage control statement related to permission or prohibition for adata usage and the duty that may be required to be performed under a permission. It is composed of a $type_i = \{permission, prohibition, duty\}, sub_i \in S, act_i \in A, res_i \in Res$ and $C_i \subset C$ that are respectively denoted as rule type, subject, action, resource, and the set of conditions so that Formula 4.2 is satisfied.

$$\begin{aligned} r_i = (type_i, sub_i, act_i, res_i, C_i), i = 1, ..., N \\ C_i = conj, j = 1, ..., M \end{aligned} \tag{4.2}$$

Where M is the number of conditions.

The components of a rule are described below.

- **Rule type** $type_i$: It refers to whether the usage control statement is a permission, prohibition, or duty. As already indicated in Section 4.2.1, the scope of our approach is limited to the analysis of DUC policies that support the heterogeneity of conditions. Thus, our work here only analyses permissions and prohibitions. Duties are the other feature introduced in DUC policies that further affects the policy quality. Analysing the impact of the duties in the quality of DUC policies is a very interesting line of research, but the scope of our approach is focused on the heterogeneity of conditions. Therefore we consider that the analysis of the quality of DUC policies taking into account the permission and prohibition rules is the primary goal in this line.

- **Subject** sub_i: It represents a participant within a data usage relationship. This may have the role of assignee or assigner. The assigner is responsible for implementing r_i, while the assignee is responsible for its enforcement. Thus, the assigner has no impact on policy quality. Therefore, sub_i always refers to the assignee in this study.

- **Action** act_i: It describes the activity that a sub_i is permitted or prohibited to perform.

- **Resource** res_i: It represents the target digital content to which a r_i applies.

- **Conditions** C_i: They describe the specifications under which r_i applies. Each $con_j \in C_i$ is in the form of a Boolean expression, and thus composed of two operands ($leftOperand_j$ and $rightOperand_j$) compared by an operator ($operator_j$) which results in either true or false. Thus, we define condition con_j according to the Formula 4.3.

$$con_j = \{leftoperand_j, operator_j, rightOperand_j\} \tag{4.3}$$

Based on this generic structure for expressing DUC policies, the rest of our work is presented below.

4.4 Definitions and Axioms

According to the generic structure for DUC policies described in the previous section, this section includes the definitions and axioms assumed in our approach for the analysis of DUC policies.

Based on the AC research literature, we introduce new definitions and axioms to deal with the heterogeneity of conditions that is introduced in DUC policies. These are indicated below.

- To handle rules refined not only through one condition, but also, multiple conditions, we define the concept of composite and atomic condition rules and state how the former is transformed into the latter.

- To manage conditions applying to different domains, we define the dependency relationship of conditions.

- To establish the scope of conditions, we detail the whitelisting and blacklisting approach for the definition of conditions and state their impact.

We have also defined the concept of time complexity. It is an important additional aspect to measure the performance provided to analyse the policies. This performance will have an impact on the overall performance of the DUC solution operation. As mentioned before, the performance of DUC solutions may render its adoption for specific data sharing scenarios. Therefore, the time complexity is an interesting aspect to evaluate the feasibility of our approach for policy analysis.

All the definitions and axioms are described below.

Definition 1. Atomic Condition Rule: if a rule r_1 is refined by only one condition, such that $M = 1$, we define that r_1 is an atomic condition rule.

Definition 2. Composite Condition Rule: if a rule r_1 is refined by more than one condition, such that $M > 1$, we define that r_1 is a composite condition rule.

Axiom 1. Rule Condition Atomization: given a composite condition rule $r_1 = (type_1, sub_1, res_1, act_1, C_1 = \{con_{11}, con_{12}\})$, we can split it into atomic condition rules $r_{11} = (type_1, sub_1, res_1, act_1, \{con_{11}\})$ and $r_{12} = (type_1, sub_1, res_1, act_1, \{con_{12}\})$ maintaining the entire scope.

Thus, a set of rules R_1, which can be refined by atomic conditions and/or composite conditions, can be split into a set of atomic

condition rules R_2 that will be analysed for policy quality. Thus, hereinafter we only refer to atomic condition rules.

Definition 3. Dependency Relationship of Conditions: for conditions con_1 and con_2, we say that these conditions meet the dependency relationship of conditions $con_1 \leftrightarrow con_2$ if they apply to the same domain, such as time, so that they may overlap. Furthermore, we define them as applying to the same domain and thus, that the dependency relationship of conditions is satisfied if their corresponding $leftoperand_1$ and $leftoperand_2$ are related to the same application domain L_{ad} so that Formula 4.4 is met.

$$con_1 \leftrightarrow con_2 = (leftoperand_1 \in L_{ad} \wedge leftoperand_2 \in L_{ad}) \quad (4.4)$$

For example, two conditions con_1 and con_2 which refine r_1 and r_2 apply to time if their corresponding $leftoperand_1 = currenttime \in L_{time}$ and $leftoperand_2 = duration \in L_{time}$ are related to time. However, con_1 and con_3 which refine r_1 and r_3 are not dependent, for example, if $leftoperand_1 = currenttime \in L_{time}$ and $leftoperand_3 = connector \in L_{connector}$ are respectively related to time and the connector.

Definition 4. Overlapping Relationship of Conditions: for conditions con_1 and con_2, if $con_1 \cap con_2 \neq \phi$, we consider that con_1 and con_2 satisfy the overlapping relationship of conditions. This is defined in Formula 4.5

$$con_1 \cap con_2 \neq \phi \quad (4.5)$$

Definition 5. Conditions Whitelisting Approach: In this approach, policies are implemented only by means of permission rules that explicitly implement all the conditions that must be satisfied to allow data usage. This case involves that if at least one rule is not satisfied, the permission is not allowed. This leads to Axiom 2, which we define as follows.

Axiom 2. If a rule r_1 of $type_1 = permission$ is set for subject sub_1 on resource res_1 with action act_1 given a condition con_1 related to a specific application domain such that $leftoperand_1 \in L_{ad_1}$ as shown in Formula 4.6, the enforcement of r_1 for dependent but non-overlapping conditions with respect to con_1 will always lead to a prohibition.

$$r_1 = (type_1 = permission, sub_1, res_1, act_1, con_1 = (leftOperand_1 \in L_{ad_1}, operator_1, rightOperand_1)) \tag{4.6}$$

Besides that, complementary permissions such as for con_2 can be implemented on dependent and overlapping conditions so that permission is refined at $con_1 \cap con_2$.

For example, the enforcement of a rule r_1 of the permission type leads to permission from 1 January 2022 to 1 January 2023, but condition enforcement at any other time will result in a prohibition. Moreover, a permission rule r_2 can be implemented as complementary at maintenance time so that permission is only granted at maintenance time between 1 January 2022 and 1 January 2023.

Definition 6. Conditions Blacklisting Approach: following this approach data usage is granted by default, and only prohibition rules are implemented, so that if only one prohibition is satisfied, i.e., if only one condition is satisfied, data usage is blocked. From this, we define Axiom 3.

Axiom 3. If a rule r_1 of $type_1 = prohibition$ is set for subject sub_1 on resource res_1 with action act_1 given a condition con_1 related to a specific application domain such that $leftoperand_1 \in L_{ad_1}$ as shown in Formula 4.7, the enforcement of r_1 for non-overlapping but dependent conditions with respect to con_1 will lead to permission by default.

$$r_1 = (type_1 = prohibition, sub_1, res_1, act_1, con_1 = (leftOperand_1 \in L_{ad_1}, operator_1, rightOperand_1)) \tag{4.7}$$

However, complementary prohibitions such as for con_2 can be implemented on dependent but non-overlapping conditions, so that prohibition is extended to $con_1 + con_2$.

For example, the enforcement of a rule r_1 of type prohibition leads to a prohibition for a Connector A, while a rule r_2, also of the prohibition type, will complement the previous one, describing a prohibition also for Connector B. However, for all other connectors data usage is permitted.

Definition 7. Dependency Relationship of Resources: for resources res_i and res_j, if res_j is part of res_i so that $res_i \rightarrow res_j$, we say that they satisfy the dependency relationship of resources.

$$res_i \rightarrow res_j \tag{4.8}$$

Definition 8. Time Complexity: This measures the time required to execute an algorithm. It is measured as the number of times that the statements of an algorithm are executed. It is expressed using the big $O()$ annotation.

4.5 Tree Structure for Efficient Policy Analysis

As mentioned before, this thesis addresses the analysis of the policy quality for DUC policies with the goal of avoiding those situations that jeopardize the adoption of DUC and, consequently, data sharing. To this end, based on the AC research literature, we have developed an approach for the analysis of the policy quality for DUC policies.

It is worth mentioning that this approach must be efficient. By efficient we mean that all the policy quality issues have to be detected. Otherwise, aforementioned security and performance issues

could appear. In turn, it should be done in the shortest possible time so that the overall performance of DUC is not affected.

This section presents the approach that we define to analyse the policy quality for DUC policies. For a more comprehensive description of our approach, this section is divided as follows:

- Efficient Policy Analysis: As reviewed in Section 4.2.1, our approach focuses on the analysis of DUC policies that support the heterogeneity of conditions. Some initial concerns addressed in the AC research literature to efficiently analyse AC policies are presented. In turn, they are extended to consider the heterogeneity of conditions.

- Tree structures: As indicated in Section 4.2.3, our approach analyse the policy quality for DUC policies following the graph-tree based modelling approach. In particular, considering those concerns related to the efficient policy analysis, tree structures are built for resources that refer to the data to be controlled and the rules implemented on them that compose the policies. Based on these structures, the quality of the policies is analysed. The most significant features of the tree structures are described.

4.5.1 Efficient Policy Analysis

As indicated in Section 4.2.2, this thesis focuses on the analysis of consistency and minimality policy quality requirements for DUC policies. In particular, our approach detects inconsistencies and redundancies among the rules that compose the policies. To these inconsistencies and redundancies we refer to as conflicts. These conflicts may appear for a particular resource or two resources that satisfy the dependency relationship of resources also denoted as dependent resources. In this regard, we differentiate between common and dependent resource conflicts.

For N atomic condition rules, conflicts can be detected by brute force by making a total of $N * (N - 1)$ comparisons between each of the rules. However, that results in poor performance for two reasons. These are the irrelevant analysis of rules that never lead to a conflict and unnecessary conversion of heterogeneous conditions.

4.5.1.1 Irrelevant Analysis of Rules that Never lead to Conflict

For two rules, r_1 and r_2, inconsistencies and redundancies appear only for the same subject and action, so $sub_1 = sub_2 \wedge act_1 = act_2$. Furthermore, they may appear for common resources $res_1 = res_2$ as well as for dependent resources $res_1 \rightarrow res_2$. As a result, only rules satisfying Formula 4.9 and Formula 4.10 should be analyzed for common and dependent resource conflicts respectively.

$$sub_i = sub_j \wedge act_i = act_j \wedge res_i = res_j \tag{4.9}$$

$$sub_i = sub_j \wedge act_i = act_j \wedge res_i \rightarrow res_j \tag{4.10}$$

These rules that may potentially lead to conflicts are denoted below as relevant rules.

In DUC, with the refinement of rules through a wide set of conditions, not only does the number of rules increase considerably but rules for a subject and action also apply to different condition application domains which are non-dependent and thus never lead to a conflict, so their analysis is irrelevant. From this, we define Axiom 4.

Axiom 4. Efficient analysis of DUC relevant rules. To detect common and dependent resource conflicts in DUC, only relevant rules that satisfy Formula 4.11 and Formula 4.12 respectively, which introduce the dependency relationship of conditions, should be analyzed.

$$sub_i = sub_j \wedge act_i = act_j \wedge res_i = res_j \wedge con_i \rightarrow con_j \quad (4.11)$$

$$sub_i = sub_j \wedge act_i = act_j \rightarrow res_i = res_j \wedge con_i \rightarrow con_j \quad (4.12)$$

In this way, conflicts are efficiently detected. While all the inconsistencies and redundancies are detected between relevant rules, unnecessary analysis of rules that never lead to a conflict is avoided.

4.5.1.2 Unnecessary Conversion of Heterogeneous Conditions

In the AC context, conditions are always homogeneously defined. For example, time is always expressed as a range. Therefore, no conversions are needed for conditions to be made comparable. In DUC, by contrast, the conditions that apply to the same domain may be heterogeneously expressed. Time conditions are a case in point. Time may be expressed as a range or as an event in time. As a result, direct comparisons cannot be made, and condition conversions are required. However, conversions should not be made for every comparison. On this basis, we define Axiom 5.

Axiom 5. Efficient conversion of heterogeneous conditions. Based on the dependency relationship of conditions, reference formats should be established for each condition application domain so that heterogeneous conditions are converted once to make them comparable.

Thus, conflicts will be efficiently detected. While all inconsistencies and redundancies can be detected between relevant rules based on comparable conditions, multiple conversions are avoided.

4.5.2 Tree Structures for Resources and Policies

The following subsections present the tree structures for resources and policies that, built based on Axiom 4 and Axiom 5, enable the efficient detection of common and dependent resource conflicts.

4.5.2.1 Resource Tree

The Resource Tree (RT), represented in Figure 4.1 through its class diagram, is intended to allow a quick search of the rules defined for a resource or dependent resources. To this end, it structures the resources available in a particular scenario based on the dependency relationship of resources and stores for each resource the policies defined for it.

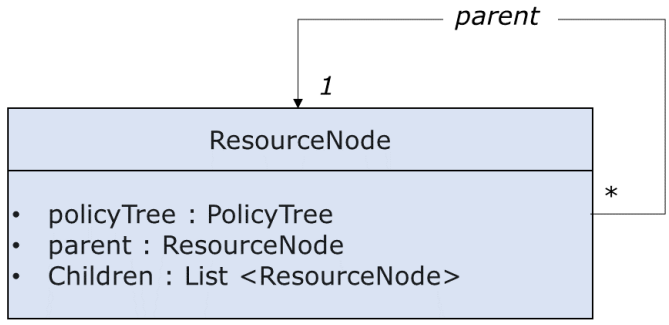

Figure 4.1: Resource tree class diagram.

Each node of the tree denoted as *resourceNode* refers to a resource. Each *resourceNode* contains its parent *resourceNode* and may contain multiple child *resourceNodes* satisfying the dependency relationship. Each *resourceNode* also contains the rules defined for it in the Policy Tree (PT).

4.5.2.2 Policy Tree

From the rules defined for a resource or dependent resources previosuly searched through the RT, the PT enables the quick search of relevant rules that may lead to a common or dependent resource conflict. To this end, it structures the rules defined for a resource.

The PT is composed of 5 types of nodes: the resource itself *res*, the actions $act_i, i = 1, ..., A$ where A is the number of actions, the subjects $sub_i, i = 1, ..., S$ where S is the number of subjects, the condition application domains $L_{ad_i}, i = 1, ..., L$ where L is the number of condition application domains, and the conditions $con_i, i = 1, ..., C$ where C is the number of conditions that refine permissions and prohibitions per application domain. Figure 4.2 represents the PT of a *resourceNode*.

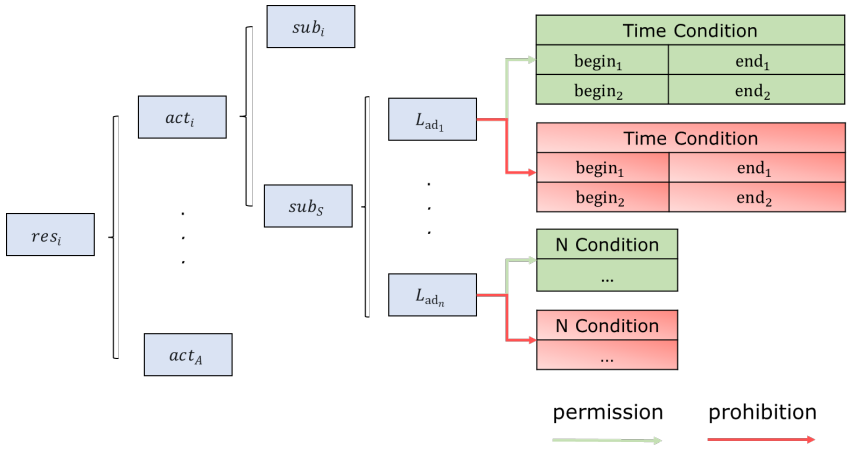

Figure 4.2: Tree structure of policy tree.

The process of building the PT for each *resourceNode* is as follows. In the first instance all the rules included within policies are sequentially analyzed in search of common resource conflicts. In this process, each rule is analyzed in relation to every relevant rule already stored in the PT of the corresponding *resourceNode*. Then the rule is stored in that PT. In this way, the following rules can be compared with that rule to check for common resource conflicts in further

analysis. Once all the rules have been analyzed, dependent resource conflicts can be checked for throughout the tree structure. To enable policy analysis to take place efficiently, the rules are structured in the PT following a set of guidelines.

First, to detect dependent resource conflicts, when common resource conflicts are detected and all the rules are stored in the PT of each *resourceNode*, searches are made from the root *resourceNode* through the RT branches sequentially comparing the resulting permissions or prohibitions implemented for each subject sub_i on each action act_i with those implemented for dependent resources. Therefore, to run this search avoiding redundancies, starting from the root node res, we built the PT by classifying the rules based on the actions act_i and subjects sub_i to which they refer. Given that there are typically fewer actions than subjects, and to reduce the number of branches and improve the search efficiency, the resource node res has one branch for each node related to the action act_i involved in data usage, but each action node act_i has one branch for each node related to the subject sub_i to which the rule refers.

Second, with the aim of avoiding the irrelevant analysis of conditions that never lead to a conflict, we classify the rules according to the dependent relationship of conditions in application domains L_{ad_i}. Specifically, in the PT each sub_i has a set of branches each of which contains a child node with one condition application domain L_{ad_i} where the rule applies. These condition application domains are identified from each $leftoperand$ that may describe a condition. Furthermore each L_{ad_i} node has two branches depending on the type of the rule implemented. Finally, each of these branches has the list of conditions under which each type of rule applies for the corresponding application domain L_{ad_i}. The PT is represented by a hashmap. A hashmap is a collection of key-value pairs. While a node of the PT represents the key, the corresponding child nodes the values. We consider that this is the most suitable approach to represent the PT as only requires *O(1)* to put and retrieve values for a key. Thus, for common resource conflicts it only takes $O(1)$ to search for relevant rules on a specific application domain L_{ad_i}. For dependent resource

conflicts, the analysis is performed individually comparing the resulting relevant permissions and prohibitions for two dependent resources for each subject sub_i, action act_i, and condition application domain L_{ad_i}. Therefore, taking at most $O(SxAxL_{ad})$.

Finally, to avoid unnecessary conversion of heterogeneous conditions, we store the lists of conditions for each application domain through reference formats. For example, time-related conditions are stored as time intervals regardless of whether they are expressed in that term or, for example, as an event in time, while user-related conditions are stored as a list of users regardless of whether they are explicitly expressed as such or implemented through a role membership. Consequently, every time a rule is analyzed, the condition that refines it is only converted once and unnecessary conversions among stored heterogeneous conditions are avoided. This improves performance in comparing conditions.

4.6 Conflict Assessment Method

In the previous section we have presented our approach based on tree structures to detect and compare all the relevant rules that may lead to common or dependent resource conflicts providing a good performance.

In this section, we present the assessment method to detect common and dependent resource conflicts comparing relevant rules. Based on the methods presented in the AC research literature, we consider the impact that conditions have on the scope of the rules if they are defined through the whitelisting or blacklisting approach.

4.6.1 Common Resource Conflicts

Below, we define how inconsistencies and redundancies appear between two relevant rules defined for a common resource. A distinc-

tion is made depending on whether the rules are defined following the whitelisting or blacklisting approach.

4.6.1.1 Whitelisting

For two relevant rules r_i and r_j when these are defined following the whitelisting approach, we state that two non-overlapping conditions such that $con_i \cap con_j = \phi$ lead to an inconsistency when Formula 4.13 is satisfied. This is because two non-overlapping conditions will never be satisfied at the same time.

$$\begin{aligned} type_i = type_j = PERMISSION \wedge sub_i = sub_j \wedge \\ act_i = act_j \wedge res_i = res_j \wedge con_i \cap con_j = \phi \end{aligned} \tag{4.13}$$

Also, if one condition is encompassed by another such that $con_i \cap con_j = con_i$ or $con_i \cap con_j = con_j$ a redundancy appears when Formula 4.14 is satisfied. One condition encompasses the other one, so the first one is unnecessary because it is less restrictive.

$$\begin{aligned} type_i = type_j = PERMISSION \wedge sub_i = sub_j \wedge \\ act_i = act_j \wedge res_i = res_j \wedge \\ (con_i \cap con_j = con_i \vee con_i \cap con_j = con_j) \end{aligned} \tag{4.14}$$

As shown in Figure 4.3, for the relevant rules r_1, r_2, r_3, and r_4 that satisfy the dependent relationship of conditions such that $leftoperand_{1,2,3,4} \in L_{time}$, con_2 refines con_1. However, while con_1 and con_3 lead to an inconsistency as $con_1 \cap con_3 = \phi$ because permission would never be granted, con_1 and con_4 lead to a redundancy because $con_1 \cap con_4 = con_4$ and permission would only be granted for con_4.

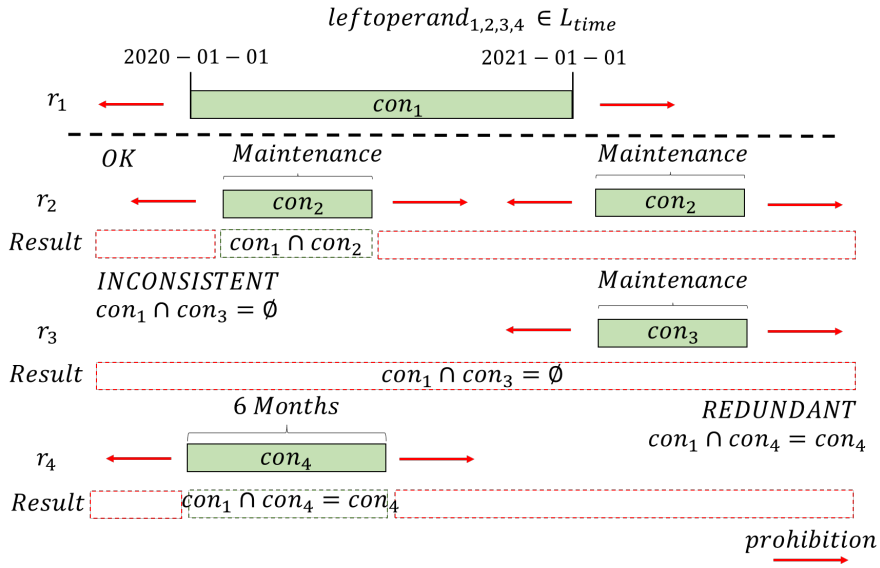

Figure 4.3: Common resource conflict detection. Whitelisting approach.

4.6.1.2 Blacklisting

In this case, for two relevant rules r_i and r_j implemented through the blacklisting approach, we say that a common resource conflict appears always as a redundancy if a condition such as con_i encompasses the other condition con_j, or viceversa so that Formula 4.15 is satisfied. The less restrictive prohibition is already encompassed by the most restrictive one.

$$\begin{aligned} type_i = type_j = PROHIBITION \wedge sub_i = sub_j \wedge \\ act_i = act_j \wedge res_i = res_j \wedge \\ (con_i \cap con_j = con_i \vee con_i \cap con_j = con_j) \end{aligned} \tag{4.15}$$

As shown in Figure 4.4, for the relevant rules r_1, r_2, and r_3, which satisfy the dependent relationship of conditions such that $leftoperand_{1,2,3} \in L_{con}$, while con_1 complements con_2, a conflict

arises as a redundancy when a condition con_2 encompasses con_3. This is because con_3 is not necessary as it is already implemented within con_2.

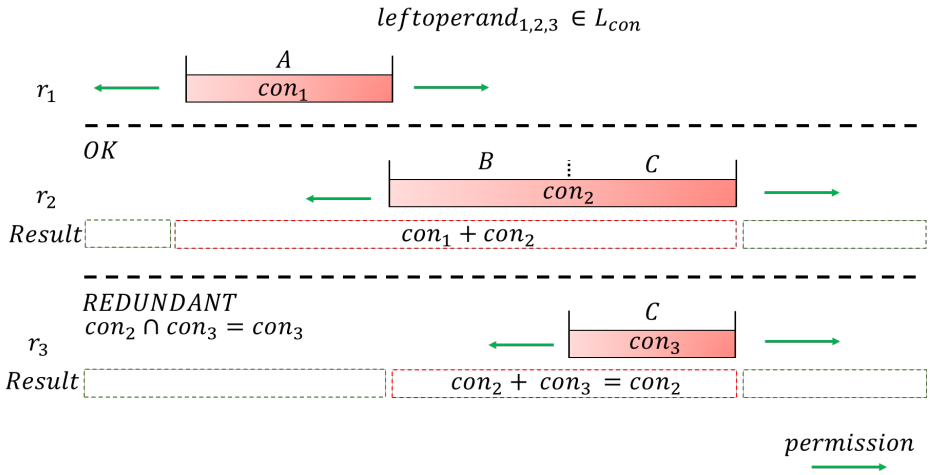

Figure 4.4: Common resource conflict detection. Blacklisting approach.

Within this approach, we consider that prohibition rules can be combined with permission rules so that refinements are introduced to the permissions granted. Thus, relevant rules r_i and r_j implemented through the whitelisting and blacklisting approaches respectively do not lead to a conflict if the prohibition refines the permission to some extent so that $con_i \cap con_j \subset con_i$.

On the one hand we define that a redundancy appears that satisfies Formula 4.16 when the two rules do not overlap so that $con_i \cap con_j = \phi$. This is because the prohibition rule is redundant with respect to the prohibition already set by the permission rule on dependent but non-overlapping conditions.

$$\begin{aligned} type_i = PERMISSION \wedge type_j = PROHIBITION \wedge \\ sub_i = sub_j \wedge act_i = act_j \wedge res_i = res_j \wedge \\ con_i \cap con_j = \phi \end{aligned} \tag{4.16}$$

On the other hand, we define that an inconsistency appears that satisfies Formula 4.17 if the prohibition rule makes an exception to the entire permission so that $con_i \cap con_j = con_i$ and the permission becomes invalid.

$$type_i = PERMISSION \wedge type_j = PROHIBITION \wedge \\ sub_i = sub_j \wedge act_i = act_j \wedge res_i = res_j \wedge \\ con_i \cap con_j = con_i \tag{4.17}$$

As shown in Figure 4.5, for the relevant rules r_1, r_2, r_3, r_4 refined through dependent conditions, a refinement is correctly made for r_1 when r_2 is implemented. Whereas r_4 leads to a redundancy as the prohibition is already made by r_1, r_3 results in an inconsistency because it makes the permission already implemented invalid regardless of the condition.

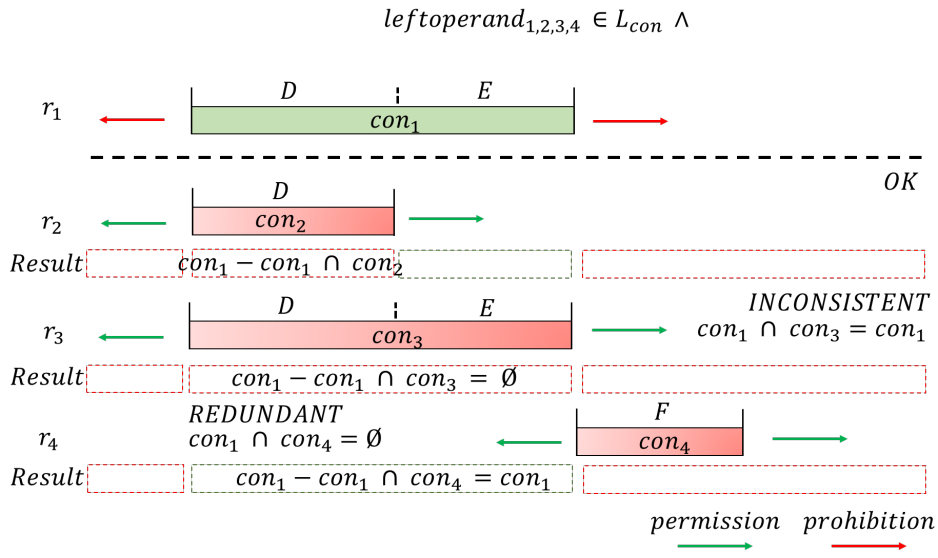

Figure 4.5: Common resource conflict detection. Whitelisting and blacklisting approach.

4.6.2 Dependent Resource Conflicts

To detect conflicts between dependent resources, we define the transmission relationship of usage authority. It establishes that for a resource, resulting permissions or prohibitions from a set of relevant rules defined for the same subject and action on a condition application domain must be less restrictive rather than for the ones from the child resources.

The following subsections present the concept of the transmission relationship of usage authority. A distinction is also made depending on whether the rules are defined following the whitelisting or blacklisting approach. Based on the transmission relationship of usage authority, we also define how inconsistencies appear.

4.6.2.1 Whitelisting

For the whitelisting approach, we define the transmission relationship of usage authority as per Figure 4.6.

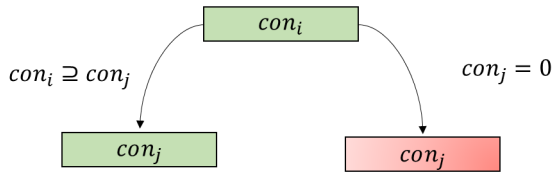

Figure 4.6: Transmission relationship of usage authority. Whitelisting approach.

Based on the resulting permissions from a set of relevant rules defined for two dependent resources for a subject and action on a specific application domain, it can be said that if a permission is defined on an upper resource as shown in Figure 4.6, the transmission relationship of usage authority is satisfied if a more restrictive permission such that $con_i \supseteq con_j$ (left branch) or a non-refined prohibition

is implemented on child resources (right branch).

If a less restrictive permission $con_i \subseteq con_j$ satisfying Formula 4.18 is defined on child resources, an inconsistency appear.

$$type_i = type_j = PERMISSION \wedge sub_i = sub_j \wedge act_i = act_j \wedge res_i \rightarrow res_j \wedge con_i \subseteq con_j \tag{4.18}$$

Also an inconsistency appears if a prohibition is defined on child resources satisfying Formula 4.19.

$$type_i = PERMISSION \wedge type_j = PROHIBITION \wedge sub_i = sub_j \wedge act_i = act_j \wedge res_i \rightarrow res_j \wedge con_j \neq \phi \tag{4.19}$$

4.6.2.2 Blacklisting

We define the transmission relationship of usage authority for the blacklisting approach as per Figure 4.7.

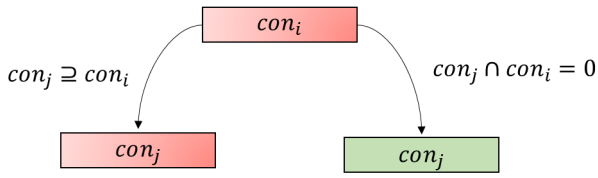

Figure 4.7: Transmission relationship of usage authority. Blacklisting approach.

In particular, also based on the results obtained for two dependent resources from a set of relevant rules on a subject, action and a specific application domain, if a prohibition is defined on an upper

resource as shown in Figure 4.7, it can be said that the transmission relationship of usage authority is satisfied if either a more restrictive prohibition is expressed on child resources such that at least the upper resource conditions are also included satisfying $con_i \subseteq con_j$ (left branch), or a permission is explicitly defined on a condition where a prohibition has not been explicitly expressed that satisfies $con_i \cap con_j = \phi$ (right branch).

If the prohibition set by the parent resource is not included by child resources satisfying Formula 4.20, an inconsistency appear.

$$\begin{aligned} type_i = type_j = PROHIBITION \wedge sub_i = sub_j \wedge \\ act_i = act_j \wedge res_i \rightarrow res_j \wedge con_i \nsubseteq con_j \end{aligned} \tag{4.20}$$

Also an inconsistency appears if a permission on child resources overrides somehow the prohibition set by the parent resource satisfying Formula 4.21.

$$\begin{aligned} type_i = PROHIBITION \wedge type_j = PERMISSION \wedge \\ sub_i = sub_j \wedge act_i = act_j \wedge res_i \rightarrow res_j \wedge \\ con_i \cap con_j \neq \phi \end{aligned} \tag{4.21}$$

4.7 The CAPA Algorithm

The CAPA algorithm following the graph/tree-based modelling approach has been developed to analyse the consistency and minimality policy quality requirements for context-aware DUC policies that support the heterogeneity of conditions.

In particular, CAPA analyses context-aware DUC policies defined making use of the general structure described in Section 4.3. From the analysis of the policies, it detects conflicts in the form of inconsistencies and redundancies. Based on the tree structures presented

in Section 4.5, those relevant rules that may lead to a conflict can be identified and compared. Applying the conflict assessment method presented in Section 4.6, conflicts are detected comparing relevant rules.

This section describes the policy analysis procedure of CAPA and measures its theoretical performance in terms of time complexity. In this regard, Figure 4.8 represents the flow diagram of CAPA.

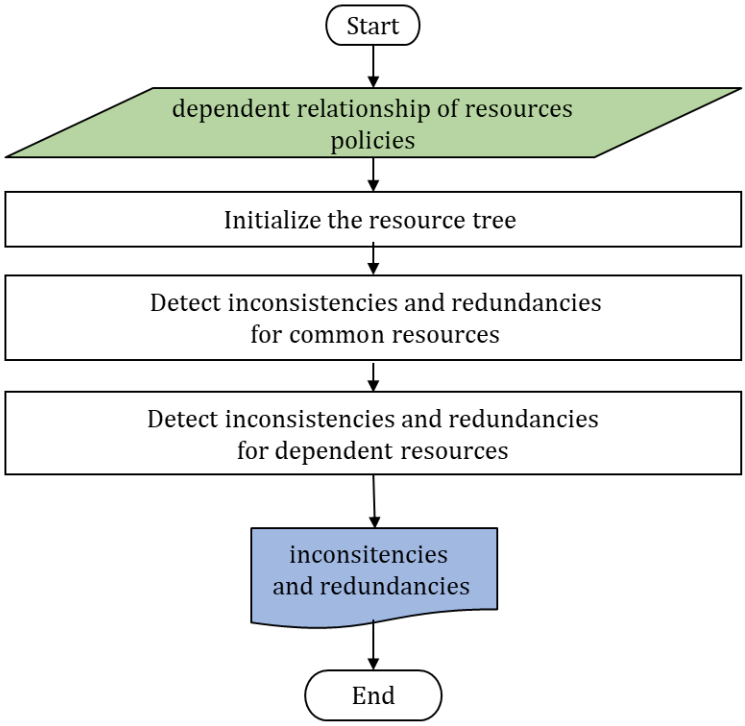

Figure 4.8: CAPA algorithm flow diagram.

For the execution of CAPA, the configuration of the dependency relationship of resources available in a particular scenario and the policies to be analysed for the policy analysis procedure are required. The policy analysis procedure is composed of the following processes:

- Initialize the resource tree: Based on the dependency relationship of resources, it structures the resources in the RT and returns the RT with the PT of each node of the RT denoted as *resourceNode* empty.

- Detect inconsistencies and redundancies for common resources: Based on the policies to be analysed and the RT initialized from the previous process, CAPA consecutively analyses each rule of each policy and detects conflicts in the form of inconsistencies and redundancies from common resources. Each time a rule is analysed, it is stored in the PT of the corresponding *resourceNode* so that it is considered for further analysis. Once all the rules have been analysed, the RT is returned with all the rules stored and structured in the PT of each *resourceNode*.

- Detect inconsistencies and redundancies for dependent resources: Based on the RT from the previous process containing all the rules of the policies in each PT of the corresponding *resourceNode*, it detects dependent resource conflicts through a backtracking process made from the root *resourceNode* throughout all the branches.

The processes that initialize the RT and detect inconsistencies and redundancies for common and dependent resources illustrate the impact of our contributions on the policy analysis procedure. The following subsections describe them in depth. Furthermore, for every process the time complexity has been evaluated to highlight its performance.

4.7.1 Initialize the Resource Tree

From the information about the dependency relationship of resources required in CAPA for the policy analysis procedure, this process structures the resources in the RT and returns the RT with the PT of each *resourceNode* empty. Figure 4.9 represents its flow diagram.

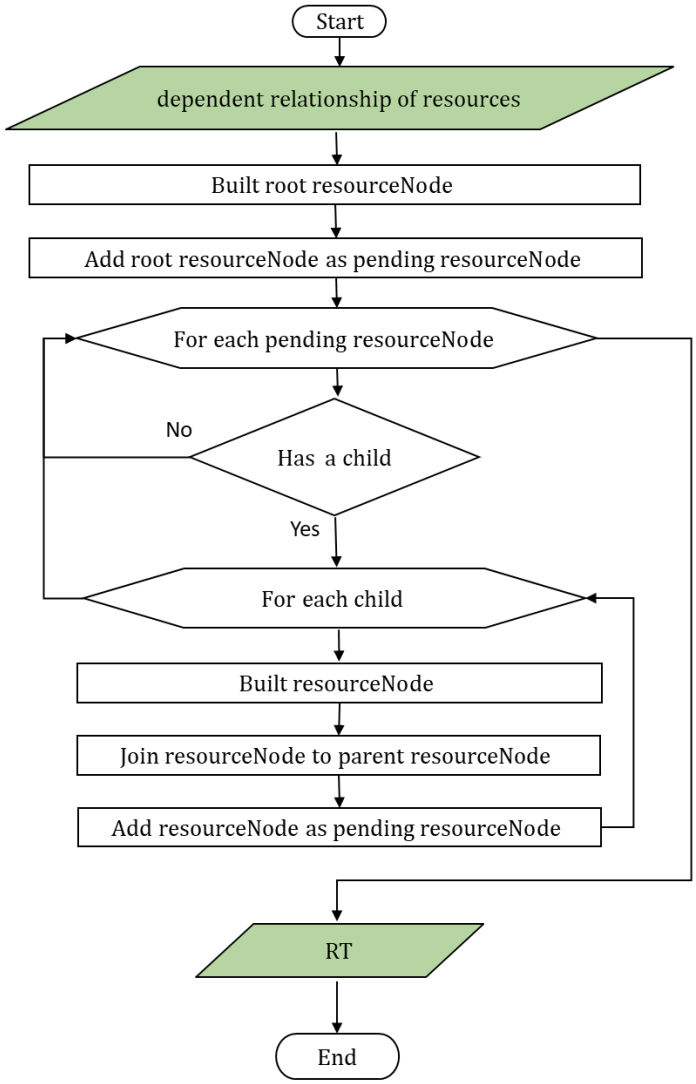

Figure 4.9: Process to initialize the resource tree.

The dependency relationship of resources is stored in a hashmap. A hashmap only requires $O(1)$ to retrieve values from a key. We consider that using a hashmap is a good approach to store a resource as the key and the list of resources that satisfy the dependency relationship as the value. From the hashmap, the root *resourceNode* is built for the root resource. From the root *resourceNode*, the RT is built by sequentially analyzing in the hashmap the resources that satisfy the dependency relationship. In particular, children resources are searched from the hashmap. For each of them, a *resourceNode* is built and joined to the parent *resourceNode*. Furthermore, they are stored to be analyzed in the following iteration. As a result, the RT is built and returned with the PT of each *resourceNode* empty.

Regarding the time to initialize the resource tree it is easy to deduce that for a set of resources *Res*, it takes *O (Res)* to initialize the resource tree.

4.7.2 Detect Inconsistencies and Redundancies for Common Resources

Starting with the RT with the PT of each *resourceNode* empty returned by the previous process, this process consecutively analyses each rule of each policy to detect inconsistencies and redundancies for common resources. Once a rule is analysed, it is stored in the PT of the corresponding *resourceNode*. Thus, it is considered in the analysis of the following rules. As a result, the RT with each rule of each policy stored in the PT of the corresponding *resourceNode* is returned. Figure 4.10 shows its flow diagram.

From the RT, for each rule of each policy, the *resourceNode* that belongs to the resource defined in the rule is found. From the rule, the application domain of the condition that refines it is obtained. Furthermore, the condition is retrieved in its reference format. Based on the condition application domain and reference format, inconsistencies and redundancies are detected for common resources between the rule and the relevant rules already stored in the PT of the *re-*

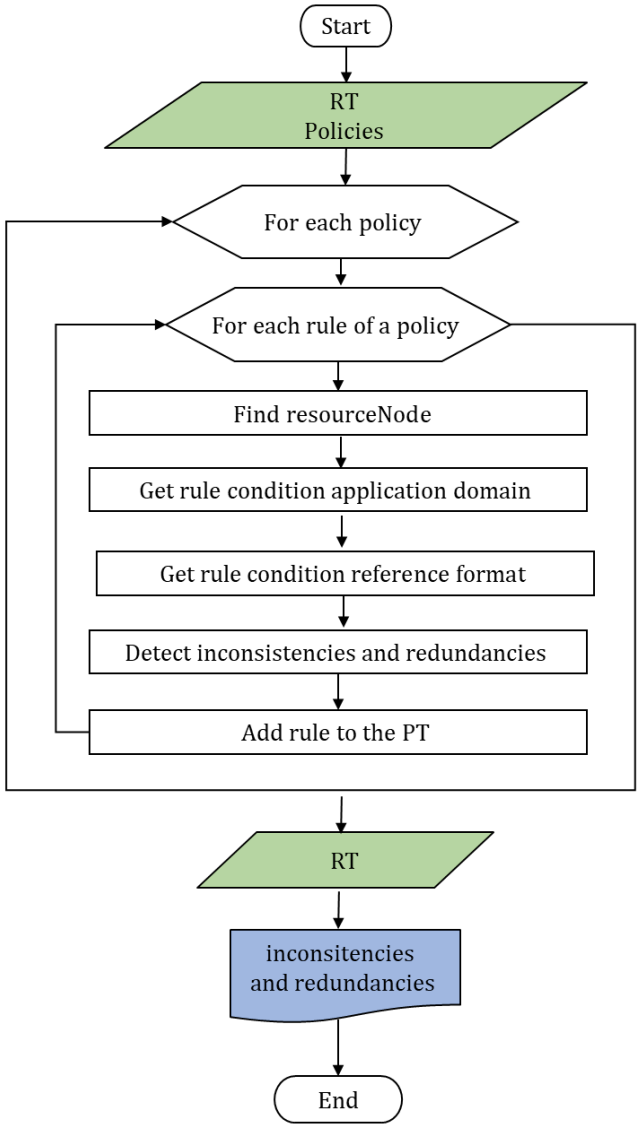

Figure 4.10: Process to detect inconsistencies and redundancies for common resources.

sourceNode found. Moreover, the rule with the condition in the reference format is added to the PT of the corresponding *resourceNode* so that it can be used in the analysis of the following rules. As a result, the RT with all the rules stored in every PT of the corresponding *resourceNode* is retrieved.

The process that finds a *resourceNode* in the RT and the one that detects inconsistencies and redundancies for common resources become complex. Thus, they are further analysed in the following subsections. In the opposite way, while the processes of getting the condition application domain and reference format return a value from a hashmap, the process that adds the rule to the PT of the *resourceNode* puts a value on a hashmap. Thus, those processes imply a time complexity of *O (1)*.

4.7.2.1 Find resourceNode

Starting with the RT and a rule, this process searches and returns the *resourceNode* that belongs to the resource defined on the rule. Figure 4.11 describes its flow diagram.

The root *resourceNode* is matched with the resource defined in the rule. If it matches, the search is stopped and the *resourceNode* is returned. If not, each child *resourceNode* is retrieved and stored for further analysis.

Therefore, for a set of resources *Res* the process to find a *resourceNode* takes *O (1)* in the best case. if the rule is defined for the root *resourceNode*, and *O (Res)* in the worst case, when all the branches of the *RT* are analyzed until the *resourceNode* is found.

4.7.2.2 Detect Inconsistencies and Redundancies

This process detects and returns inconsistencies and redundancies for common resources between a rule and the *resourceNode* found in

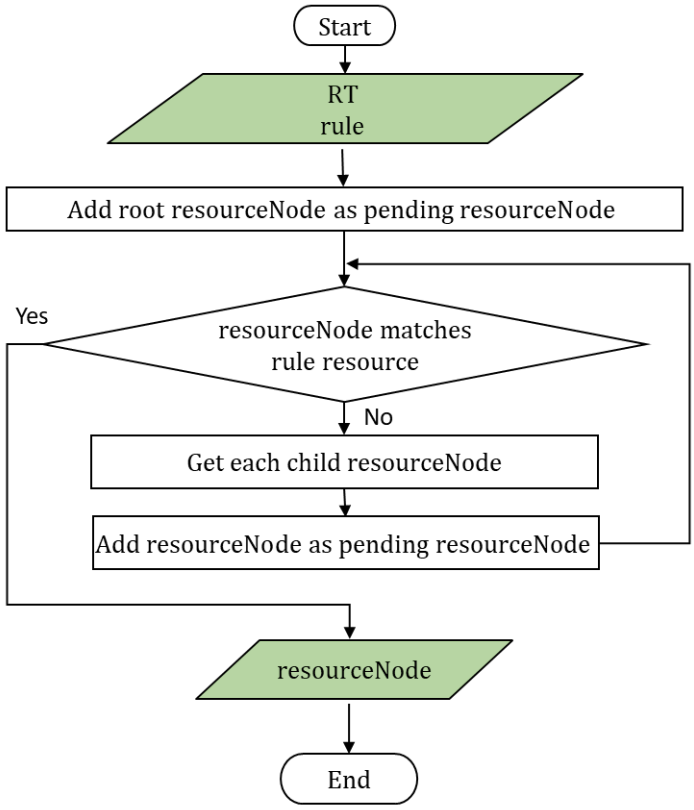

Figure 4.11: Process to find resourceNode.

the above process. It does so based on the rule condition application domain and its reference format. From the application domain, it searches for relevant rules in the *resourceNode*. Using the reference format, comparisons between conditions are made to detect inconsistencies and redundancies between the rule and the relevant rules previously searched. Figure 4.12 shows this process.

The process is represented for time-based conditions, but it can be extended to other condition application domains. Based on the action, subject and the condition application domain involved in the rule to be analysed, permission and prohibition relevant rules are

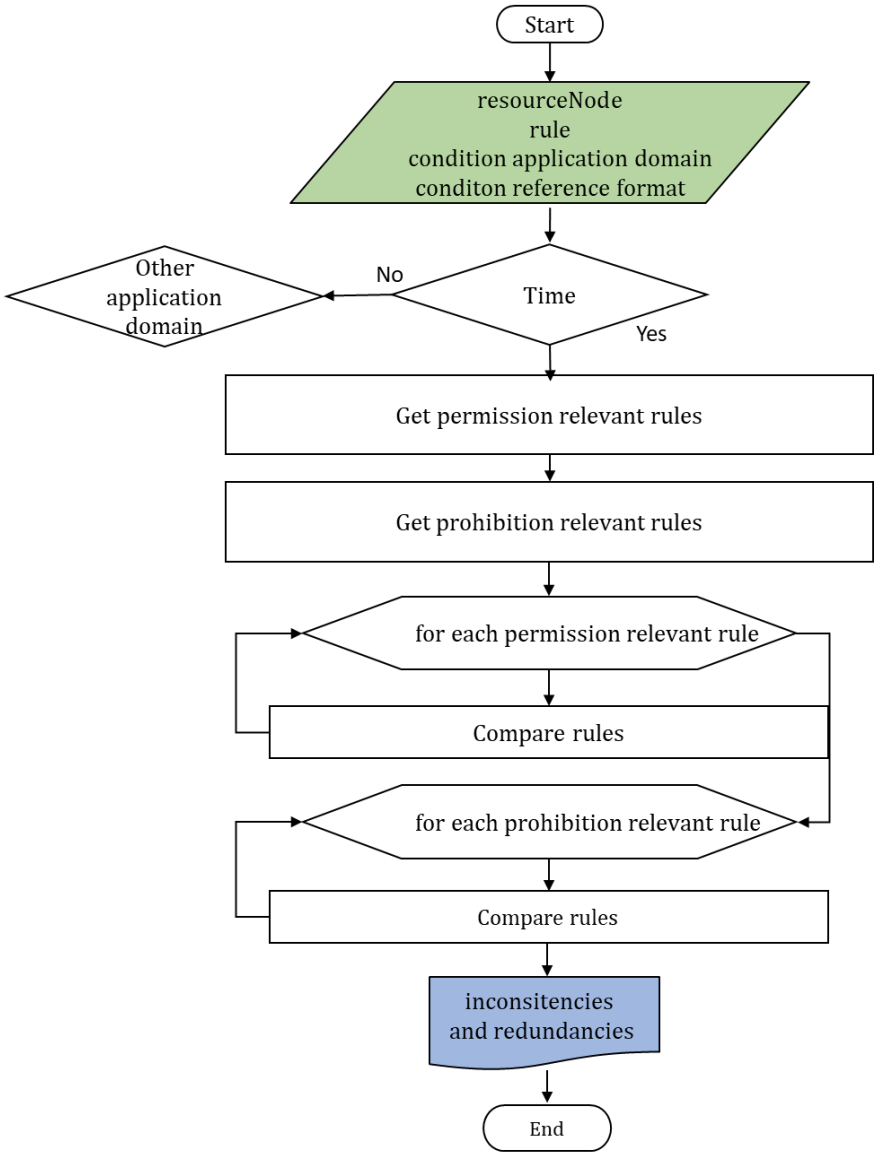

Figure 4.12: Process to detect inconsistencies and redundancies.

retrieved from the PT of the *resourceNode*. The condition that refines each relevant rule is compared with the condition of the rule in its reference format. This detects inconsistencies and redundancies.

The time complexity of this process can be analyzed in the following two complementary ways.

- Relevant rules search: Because the PT is built on a hashmap, it takes *O (1)* to retrieve relevant rules from a *resourceNode* for permissions and prohibitions.

- Rule comparison: For a set of rules R, it only requires R conversions: one for each rule. On the contrary, if done by brute force, comparing heterogeneous conditions requires one conversion for the condition that refines the rule and another for the condition of each relevant rule stored in the PT.

4.7.3 Detect Inconsistencies and Redundancies for Dependent Resources

From the RT of the previous process and with all the rules stored in each PT of the corresponding *resourceNode*, this process detects and returns inconsistencies and redundancies between dependent resources. To do so, it makes a backtracking process from the root *resourceNode* along all its branches. Figure 4.13 shows this process.

From the root *resourceNode*, it is checked whether each *resourceNode* has children *resourceNodes*. If so, the parent *resourceNode* is analysed against each child *resourceNode*. Furthermore, each child *resourceNode* is stored for further analysis along its branches. To detect inconsistencies and redundancies, a sequential search is performed over the actions of each subject that are controlled by at least one rule for the parent *resourceNode*. For each one, if the child *resourceNode* is also controlled for the same subject and action, inconsistencies and redundancies are sequentially searched for

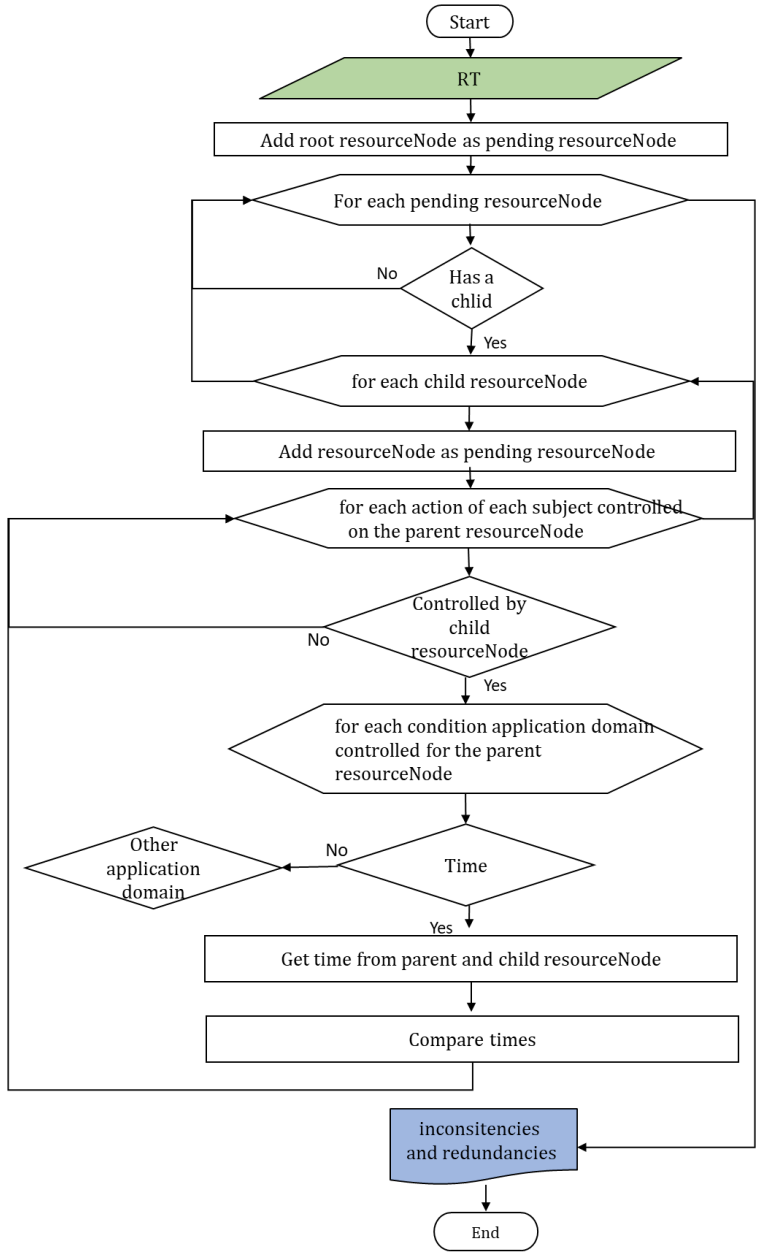

Figure 4.13: Process to detect inconsistencies and redundancies for dependent resources.

and detected for each condition application domain. Figure 4.13 represents the process for time-based conditions, but the method can be extended to all other domains. For each subject, action and condition application domain, resulting conditions defined for both the parent and child *resourceNode* are calculated and retrieved. Based on them, a comparison is made. As a result, inconsistencies and redundancies are detected.

The time complexity of this process can be analyzed in the following three complementary ways:

- Inconsistencies and redundancies search: given a resource controlled for a set of subjects S on a set of actions A and refined through L_{ad} application domains, it takes at most $O\ (\ S\ x\ A\ x\ L_{ad})$ to search for inconsistencies and redundancies with each dependent resource.

- Relevant rules search: for a *resourceNode*, it takes $O\ (1)$ to retrieve relevant rules from a parent and child *resourceNode*.

- Rule comparison: conditions are already stored in reference formats so new conversions are not required.

4.8 Conclusions

One of the critical aspects for controlling the adequate usage of the data in the DUC context are the policies. By means of the usage of policies it is possible to define restrictions on the usage of the data. Considering them, DUC solutions decide whether the data provider shares the data and if so, if the data consumer can use the data. If policies are not correctly defined, two undesired situations may arise. These are the unauthorized use of data and slow decision making on data usage. In this context, the policy quality sets basis requirements that define if policies are well-defined so that their enforcement does not lead to these security and performance issues.

Security and performance are of utmost importance for DUC solutions to promote data sharing. If an unauthorized data use happens, data sovereignty will be compromised. This put risk at the adoption of DUC solutions. In turn, if a usage decision take more time than necessary, additional delays will appear at data sharing. This may make the applicability of DUC solutions not feasible for specific scenarios. Therefore, we consider the policy quality a key aspect for DUC. However, this aspect has not yet been addressed.

In the the AC context, which constitutes the basis for UC and DUC, the requirements for ensuring the proper quality in AC policies have already been defined. These are consistency, minimality, completeness, relevance and correctness. In turn, several algorithms have been developed to analyse these requirements for AC policies following different approaches. These approaches are formal methods, model checking methods, data mining methods, graph/tree-based modelling methods and mutation testing. Following a logical approach, we consider it appropriate to address the analysis of the policy quality for DUC policies based on the AC research literature. To do so, we have considered three aspects.

First, DUC policies introduce two features with respect to AC policies. These are the heterogeneity of conditions and extended control by supporting duties. Although security and performance problems related to poor policy quality should be avoided, we consider security a more critical aspect to be ensured. This is because while security put data sharing at risk as data sovereignty is compromised, performance losses can be assumed to some extent in certain scenarios. On this basis, security is affected if permissions and prohibitions are not correctly defined. In turn, the heterogeneity of conditions makes the appearance of badly defined permissions and prohibitions more prone. Thus, we consider more critical to address the heterogeneity of conditions feature of DUC policies rather than the extended control by supporting duties. Further on, the extended control by supporting duties can be addressed.

Second, we consider consistency, minimality, relevance, completeness and correctness policy quality requirements for AC policies suitable for DUC policies. As a first step towards the analysis of the policy quality for DUC policies we consider adequate to analyse consistency and minimality. This is because it only requires a policy-based analysis. Moreover, they are the more critical ones. While unauthorized data uses mainly result from inconsistencies, longer policy enforcement times are generally caused by redundancies. To analyse relevance and completeness apart from a policy-based analysis a transaction-based analysis is required. To analyse correctness, the intended goals of the policies must be known. Thus, relevance, completeness and correctness can be afterwards addressed based on the policy-based analysis considering the additional information required.

Third, from the approaches proposed for the analysis of the policy quality for AC policies, we consider suitable for DUC only those approaches that are able to detect all the issues that do not satisfy the policy quality requirements taking into account the features introduced in DUC policies. Otherwise, policy quality will not be completely addressed. Based on this, we consider only suitable for DUC the graph-tree based modelling approach. It enables the analysis of all the policy quality requirements representing policies as tree or graph structures. As policies are defined as structures composed of different interrelated components, it is easy to represent policies as tree or graph structures. Based on tree or graph structures, searches among policies can be quickly performed detecting all the policy quality issues.

Considering these aspects, this section has presented an approach that, based on the graph/tree-based modelling approach, analyses consistency and minimality policy quality requirements for context-aware DUC policies that support the heterogeneity of conditions feature. Starting from the most significant aspects of the algorithms developed so far in this line in AC research literature, this approach presents a set of contributions that are described below.

First, considering the most significant features of the policy languages for DUC, we define and formalize a generic structure for DUC policies. Regardless of the policy language used in a DUC solution, DUC policies can be mapped to this generic structure. Our approach analyses DUC policies based on this structure. Thus, it is valid for any DUC solution.

Second, according to the generic structure for DUC policies, we present some definitions and axioms assumed in our approach for the analysis of DUC policies. Starting from those definitions and axioms assumed in the AC research literature, we introduce new ones to deal with the heterogeneity of conditions that have to be taken into account in DUC policies.

Third, based on the definitions and axioms assumed in our approach, we present initial concerns addressed in the AC research literature to efficiently analyse AC policies and we extend them to consider the heterogeneity of conditions. By efficient we mean that all consistency and minimality issues, which we respectively denote as inconsistency and redundancy conflicts, can be detected between the rules that compose the policies in the shortest possible time. This is of utmost importance. While the appearance of inconsistencies and redundancies causes an incorrect DUC operation, the policy analysis performance impacts on the overall performance of DUC. Considering these concerns, we define a tree structure on which policy analysis can be performed.

Fourth, to detect inconsistencies and redundancies between rules we propose an assessment method for conflict detection. Based on the methods proposed in the AC research literature, we have included the impact the impact that the heterogeneity of conditions has on the rules.

Following this approach, an algorithm called CAPA is presented. It is a policy analysis algorithm that based on the graph-tree based modelling approach analyses consistency and minimality policy quality requirements for context-aware DUC policies that support the heterogeneity of conditions. In particular, the policy analysis proce-

dure of CAPA is described and its theoretical performance measured in terms of the time complexity.

Chapter 5

Validation

5.1 Introduction

This chapter explains the tests carried out with the aim of analyzing the validity of the contributions of this thesis. These are the framework for the identification and assessment of main features of DUC solutions presented in Chapter 3 and the context-aware policy analysis algorithm for DUC presented in Chapter 4.

As pointed out in Section 2.2, existing DUC solutions have followed different approaches supporting different features. These are detailed in Section 2.3. In this context, we think that it is important to identify the features that DUC solutions must support to achieve data sovereignty requirements in the context of data sharing. In turn, Section 2.4 remarks that there is no framework for the assessment of DUC solutions. We consider also fundamental to identify to what extent DUC solutions ensure data sovereignty and the limitations that they present. In this regard, in Chapter 3 we propose a framework for the identification and assessment of the main features that DUC solutions must support to ensure data sovereignty. To identify the features, we have analysed how data sovereignty requirements in the context of data sharing can be satisfied based on a detailed analysis

of existing research related to AC, UC and DUC. Considering the features identified, we have proposed a framework for their assessment. The aim of this section is to analyse if this framework enables us to know to what extent DUC solutions ensure data sovereignty. To do so, the framework is applied to the most widespread DUC solutions. From the results of the application of the framework, three aspects are considered. First, whether there is any feature of the DUC solutions that cannot be assessed with the framework. Second, if all the features included in the framework can be assessed for the DUC solutions. Third, whether the framework allows us to identify the advantages and limitations of existing the DUC solutions.

As commented in Section 2.5, we consider the policy quality crucial to encourage data sharing through the adoption of DUC. The policy quality ensures that DUC policies are well-defined so that unauthorized data uses and longer policy enforcement times are avoided. The former compromises data sovereignty and jeopardize the adoption of DUC solutions. The latter includes additional delays on data sharing and makes DUC solutions not applicable for certain scenarios. Nevertheless, the policy quality for DUC has not yet been addressed. In the AC context, the policy quality is a research field that has been well addressed. The requirements that must be met to ensure the quality of AC policies have been defined. Also, several policy analysis algorithms have been developed to analyse these requirements for AC policies following different approaches. Following a logical approach, this thesis addresses the policy quality for DUC policies based on the AC research literature. In this regard, we have considered three aspects. They are described below:

- DUC policies introduce two features with respect to AC policies. These are the heterogeneity of conditions and the extended control by supporting duties. We think that unauthorized data uses which compromise data sovereignty are more critical rather than unnecessary policy enforcement which can be assumed in specific context. These security issues came from poorly defined permissions and prohibitions. In turn, the heterogeneity of conditions makes the appearance of these situa-

tions more probable. Therefore, although both new features of DUC policies should be considered for the analysis of the policy quality for DUC policies, we address the heterogeneity of conditions.

- We consider consistency, minimality, relevance, completeness and correctness policy quality requirements for AC policies adequate to establish the quality of DUC policies. To analyse consistency and minimality only a policy-based analysis is required. To analyse relevance, completeness and correctness additional information is required. Furthermore, consistency and minimality are the most critical ones. Unauthorized data uses mainly come from inconsistencies, while longer policy enforcement time due to redundancies. As a result, although all the five policy quality requirements should be analysed for DUC policies, we focus on consistency and minimality.

- From the approaches proposed for the analysis of the policy quality for AC policies we only consider suitable for DUC those that are able to detect any issue related to the policy quality requirements. Otherwise, data sharing is compromised. The graph/tree-based modelling approach is the only one capable of doing so. Furthermore, DUC policies are defined as different components interrelated. In this way, it is easy to represent policies as tree or graph structures where searches can be quickly performed. Therefore, we think that the graph/tree-based modelling approach is suitable to analyse the policy quality for DUC policies.

Based on these considerations, in Chapter 4 we present an approach to analyse consistency and minimality policy quality requirements for context-aware DUC policies that support the heterogeneity of conditions using the graph/tree-based modelling approach. Following this approach, CAPA has been developed. The aim of the test described in this section is to analyse whether our approach implemented in CAPA correctly analyses the consistency and minimality policy quality requirements for a set of context-aware DUC policies

that support the heterogeneity of conditions. To this end, an experimental assessment of CAPA is performed for a wind energy use case. From the result of the experimental assessment, three aspects are considered. First, if CAPA detects all the issues that do not satisfy the consistency and minimality policy quality requirements. Second, the performance improvement that our approach introduces in the policy analysis procedure by including our contributions to efficiently manage the heterogeneity of conditions. Third, whether the performance provided by CAPA is enough for a real use case.

5.2 DUC Assessment Framework Application

Chapter 3 presents the framework for the identification and assessment of main features of DUC solutions. This section analyses if it is valid to know to what extent DUC solutions ensure data sovereignty. To do so, we apply the framework to assess the most widespread DUC solutions. We consider that these are MYDATA and LUCON. We split the application of the framework based on the two requirements that DUC solutions must satisfy to ensure data sovereignty in the context of data sharing. These are policy implementation and policy enforcement. They are detailed below:

- Policy implementation: It means that policies defining restrictions on the usage of the data must be implemented considering the needs of the two organizations involved. The features that must be supported to achieve it are: policy specification, policy quality, policy negotiation and policy transformation.

- Policy enforcement: It states that policies must be enforced from its access to its multiple uses once it has been shared. The features that must be supported to achieve it are: decision continuity, permission/prohibition enforcement, duty enforcement and duty handling.

From the application of the framework, we analyse three aspects. First, if there is any feature of these DUC solutions that has not been considered. Second, if all the features included in the framework can be assessed for these DUC solutions. Third, if the advantages and limitations of these DUC solutions can be identified.

5.2.1 MYDATA Assessment

MYDATA is a commercial DUC solution that proposes an approach to control how data is used in a distributed architecture, monitoring and intercepting system relevant events. This section applies the assessment framework to MYDATA.

5.2.1.1 Policy Implementation

Figure 5.1 represents the approach proposed by MYDATA for policy implementation following the schema of Figure 3.4 and with the features classified by the same colors.

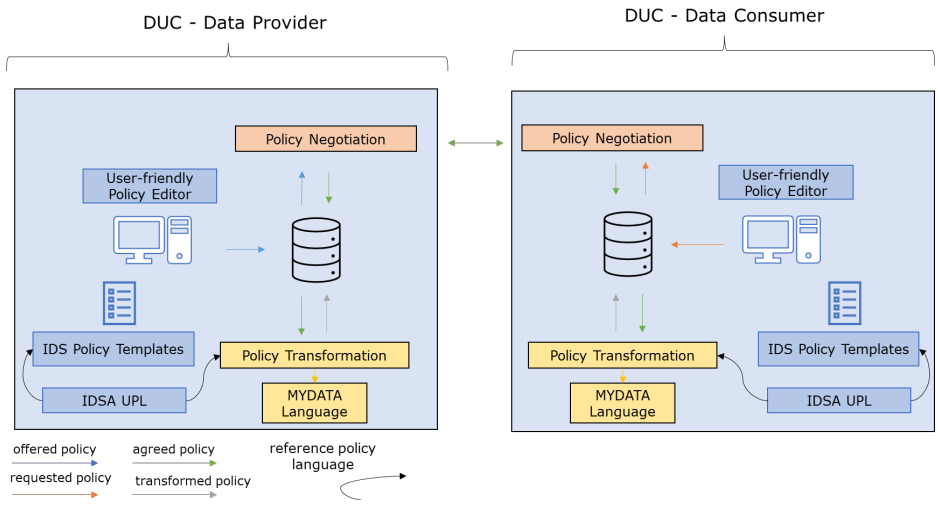

Figure 5.1: MYDATA policy implementation approach.

MYDATA proposes an approach in which the data provider and the data consumer respectively defines offered and requested policies using the IDSA UPL through the templates provided by a policy editor. Thus, policy specification is supported. Whether these policies are well-defined is not analysed. As a result, policy quality is not supported. To agree on policies between the data provider and the data consumer, there is not a policy negotiation process. Instead, the data consumer accept or deny the policies that the data provider has defined. To accept them, the data consumer must define requested policies in the same way as offered policies. Therefore, policy negotiation is not supported. These policies are transformed to policies defined using the MYDATA policy language and implemented in the data provider and the data consumer. Thus, policy transformation is supported.

In short, MYDATA supports the policy specification and policy transformation features but does not support the policy quality and policy negotiation features. Thus, the features that MYDATA support to achieve the policy implementation requirement can be assessed based on whether they are supported or not as represented in Table 5.1.

Table 5.1: MYDATA policy implementation assessment.

Feature	**Assessment**
Policy Specification	Yes
Policy Quality	No
Policy Negotiation	No
Policy Transformation	Yes

To assess the expressiveness of the policies that MYDATA handles to transform policies defined using the IDSA UPL to policies defined using the MYDATA policy language, it should be considered that MYDATA was developed to address DUC in supply networks for automotive Original Equipment Manufacturer (OEM)s. Therefore, the expressiveness of the policies defined using the IDSA UPL is oriented to this use case addressed. In turn, the expressiveness of the policies that the policy transformation feature manages too.

DUC policies are divided into permissions and prohibitions. Permissions or prohibitions define the conditions under which data usage is permitted or prohibited. Duties complement a permission to set a duty action that must be executed if data usage is permitted. In the assessment framework we state that all the features that must be supported to satisfy the policy implementation requirement can be assessed based on whether they are supported or not and for each feature supported, in this case policy transformation, based on the conditions that can be managed from permissions and prohibitions and the duty actions that can be managed from duties.

From the IDSA UPL, Table 5.2 identifies the conditions that MYDATA manages to transform permissions and prohibitions defined using the IDSA UPL to those defined using the MYDATA policy language. Table 5.3 identifies duty actions that MYDATA manages to transform duties defined using the IDSA UPL to those defined using the MYDATA policy language.

Table 5.2: Expressiveness supported by MYDATA for policy transformation regarding conditions.

ids:leftOperand	***ids:operator***
idsc:CONNECTOR	idsc:SAME_AS
idsc:USER	idsc:HAS_MEMBERSHIP
idsc:APPLICATION	idsc:EQ
idsc:POLICY_EVALUATION_TIME	idsc:TEMPORAL_EQUALS
idsc:DATE_TIME	idsc:BEFORE
idsc:ELAPSE_TIME	idsc:SHORTER_EQ
idsc:EVENT	idsc:EQ
idsc:ABSOLUTE_SPATIAL_POSITION	idsc:EQ
idsc:PAY_AMOUNT	idsc:EQ
idsc:PURPOSE	idsc:EQ
idsc:COUNT	idsc:LTEQ
idsc:SECURITY_LEVEL	idsc:EQUALS
idsc:STATE	idsc:EQUALS

Table 5.3: Expressiveness supported by MYDATA for policy transformation regarding duty actions.

ids:action	***ids:leftOperand***	***ids:operator***
idsc:INCREMENTCOUNTER		
idsc:DELETE	idsc:DELAY idsc:DATE_TIME	idsc:DUTATION_EQ idsc:BEFORE
idsc:ANONYMIZE	idsc:MODIFICATION_METHOD	idsc:DEFINES_EQ
idsc:LOG	idsc:LOG_LEVEL idsc:SYSTEM_DEVICE	idsc:DEFINES_AS idsc:DEFINES_AS
idsc:NOTIFY	idsc:NOTIFICATION_LEVEL idsc:RECIPIENT	idsc:DEFINES_AS idsc:DEFINES_AS

5.2.1.2 Policy Enforcement

MYDATA follows the event-based DUC. Therefore, it controls how data is used in a distributed architecture, monitoring and intercepting system relevant events. Section 2.3 already presents the architecture that this approach proposes based on the ABAC model architecture and the features it provides. They are summarized below.

To trigger policy enforcement, the PEP intercepts and forwards events to the PDP. Depending on the events that trigger the enforcement of policies, the PDP subscribes to different PEPs. The PDP is always subscribed to a PEP intercepting each usage request. In this way, MYDATA supports decision continuity. The PDP is also subscribed to different PEPs intercepting the events that trigger the execution of duties. As a result, MYDATA supports duty handling. Each time an event is intercepted by the PEP it is forwarded to the PDP. The PDP enforces the policies. In particular, triggered by a usage request, for each permission or prohibition, while the PDP evaluates all the conditions that refines it, the PIP gathers context information that may be required for that purpose. Thus, permission/prohibition enforcement is supported. Furthermore, the PDP

also triggers the execution of duties to the PXP. Therefore, duty enforcement is supported.

In short, MYDATA supports the permission/prohibition enforcement, duty enforcement, decision continuity and duty handling features. Thus, the features that MYDATA support to achieve the policy enforcement requirement can be assessed based on whether they are supported or not as represented in Table 5.4.

Table 5.4: MYDATA policy enforcement assessment.

Feature	**Assessment**
Permission/Prohibition Enforcement	Yes
Duty Enforcement	Yes
Decision Continuity	Yes
Duty Handling	Yes

In the assessment framework we state that all the features that must be supported to satisfy the policy enforcement requirement can be assessed based on whether they are supported or not. In turn, for the permissions/prohibition and duty enforcement an additional assessment should be performed. For the permission/prohibition enforcement feature, we state that the conditions that can be evaluated from permissions and prohibitions should be identified. For the duty enforcement feature, we define that the duty actions that can be executed from duties should be considered.

From the IDSA UPL, Table 5.5 identifies all the conditions that can be evaluated by MYDATA to enforce permissions and prohibitions. Table 5.6 identifies all the duty actions that can be executed by MYDATA to enforce duties.

5.2.2 LUCON Assessment

LUCON is an open source DUC solution that controls how data is shared and processed between services based on message labeling. This section applies the assessment framework to LUCON.

Table 5.5: MYDATA permission / prohibition enforcement assessment.

ids:leftOperand	***ids:operator***
idsc:APPLICATION	idsc:EQ
idsc:POLICY_EVALUATION_TIME	idsc:TEMPORAL_EQUALS
idsc:DATE_TIME	idsc:BEFORE
idsc:EVENT	idsc:EQ
idsc:ABSOLUTE_SPATIAL_POSITION	idsc:EQ
idsc:PAY_AMOUNT	idsc:EQ
idsc:PURPOSE	idsc:EQ
idsc:COUNT	idsc:LTEQ
idsc:SECURITY_LEVEL	idsc:EQUALS
idsc:STATE	idsc:EQUALS

Table 5.6: MYDATA duty enforcement assessment.

ids:action	***ids:leftOperand***	***ids:operator***
idsc:INCREMENTCOUNTER		
idsc:DELETE	idsc:DELAY idsc:DATE_TIME	idsc:DUTATION_EQ idsc:BEFORE
idsc:ANONYMIZE	idsc:MODIFICATION_METHOD	idsc:DEFINES_EQ
idsc:LOG	idsc:LOG_LEVEL idsc:SYSTEM_DEVICE	idsc:DEFINES_AS idsc:DEFINES_AS
idsc:NOTIFY	idsc:NOTIFICATION_LEVEL idsc:RECIPIENT	idsc:DEFINES_AS idsc:DEFINES_AS

5.2.2.1 Policy Implementation

Figure 5.2 shows the approach proposed by LUCON to implement the policies that define the restrictions on the usage of the data. It is represented following the schema of Figure 3.2 and with the features classified by the same colors.

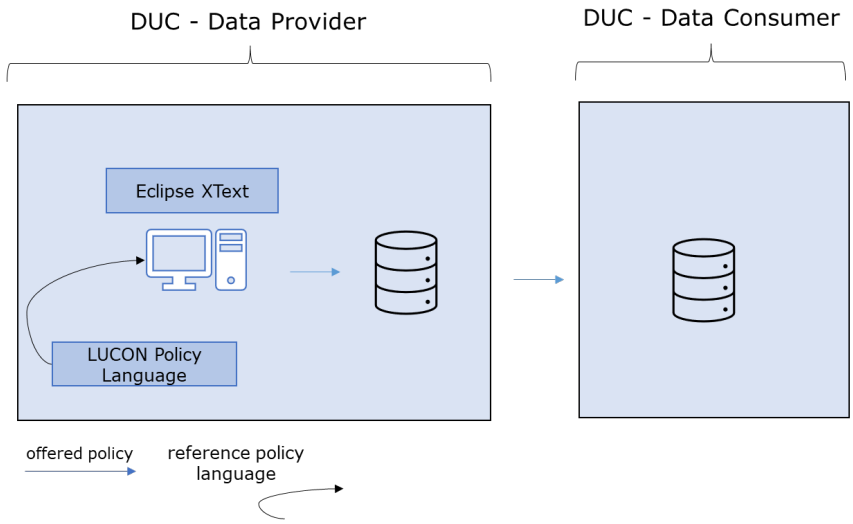

Figure 5.2: LUCON policy implementation approach.

LUCON provides a language creation framework in Eclipse XText for the data provider to define offered policies. Policies can not be defined using the reference policy language, the IDSA UPL. Instead, they are defined using the LUCON proprietary policy language, the LUCON policy language. Thus, policy specification is not supported. The quality of these policies is not analysed. Therefore, policy quality is not supported too. Requested policies can not be defined by the data consumer. Thus, the needs of the data consumer are not considered. Instead, offered policies, defined using the LUCON policy language, are assumed as agreed policies and should be propagated to the data consumer to be enforced along the period of time when data is used several times. As a result policy negotiation and policy transformation are not supported.

In summary, LUCON does not support the policy specification, policy quality, policy negotiation and policy transformation features. Thus, the features that LUCON support to achieve the policy implementation requirement can be assessed based on whether they are supported or not as shown in Table 5.7.

Table 5.7: LUCON policy implementation assessment.

Feature	**Assessment**
Policy Specification	No
Policy Quality	No
Policy Negotiation	No
Policy Transformation	No

Deepening LUCON policies, also called flow rules, they express which services can use the data based on their identifiers, their capabilities and the labels received attached to the data. Figure 2.10 presents an example of a flow rule.

Therefore, the identifier can be used to implement an application-based condition. Moreover, the labels are dynamically attached and removed through the services involved in data usage depending on how data is used. Therefore, the labels included in the flow rules can be used to specify data flow tracking related conditions. Furthermore, duties can also be implemented in the form of obligations in the flow rules.

Considering this, Table 5.8 identifies the conditions that can be included using the LUCON policy language to define permissions and prohibitions. In turn, Table 5.9 identifies the duty actions that can be included using the LUCON policy language to define duties.

Table 5.8: Expressiveness supported by LUCON policies regarding conditions.

ids:leftOperand	***ids:operator***
idsc:APPLICATION	idsc:EQ
idsc:ARTIFACT_STATE	idsc:EQ

Table 5.9: Expressiveness supported by LUCON policies regarding duties.

ids:action	***ids:leftOperand***	***ids:operator***
idsc:DELETE	idsc:DELAY	idsc:DURATION_EQ
idsc:ANONYMIZE	idsc:MODIFICATION_METHOD	idsc:EQ
idsc:LOG	idsc:SYSTEM_DEVICE	idsc:DEFINES_AS
idsc:NOTIFY	idsc:RECIPIENT	idsc:DEFINES_AS

5.2.2.2 Policy Enforcement

LUCON follows the label-based DUC. Thus, it controls how data is shared and processed between services in a distributed architecture based on message labeling. Section 2.3 details the architecture that this approach proposes, which is not based on an standardized architecture previously proposed, and the features it provides. They are described below.

To trigger the policy enforcement, LUCON relies on a routing engine. Every time the data enters or leaves a service configured on the route, the flow rules are enforced. In this way, decision continuity is supported. Every time policy enforcement is triggered, the identifier and the capabilities of the service that receives the data and the labels received attached to the data are evaluated to permit or prohibit data usage. Thus, permission/prohibition enforcement is supported. If data usage is permitted, the management of the duties is divided between LUCON and the service itself. LUCON manages logging and notification duties while additional duties are managed by the service that will use the data. In this way, duty handling and enforcement is supported.

In short, LUCON supports the permission/prohibition enforcement, duty enforcement, decision continuity and duty handling features. In this way, the features that LUCON support to achieve the policy enforcement requirement can be assessed based on whether they are supported or no as represented in Table 5.10.

Table 5.10: LUCON policy enforcement assessment.

Feature	**Assessment**
Permission/Prohibition Enforcement	Yes
Duty Enforcement	Yes
Decision Continuity	Yes
Duty Handling	Yes

From the IDSA UPL, Table 5.11 identifies all the conditions that LUCON can evaluate to enforce permissions and prohibitions. Table 5.12 identifies all the duty actions that LUCON can execute to enforce duties.

Table 5.11: LUCON permissions / prohibitions enforcement assessment.

ids:leftOperand	***ids:operator***
idsc:APPLICATION	idsc:EQ
idsc:ARTIFACT_STATE	idsc:EQ

Table 5.12: LUCON duties enforcement assessment.

ids:action	***ids:leftOperand***	***ids:operator***
idsc:DELETE	idsc:DELAY	idsc:DURATION_EQ
idsc:ANONYMIZE	idsc:MODIFICATION_METHOD	idsc:EQ
idsc:LOG	idsc:SYSTEM_DEVICE	idsc:DEFINES_AS
idsc:NOTIFY	idsc:RECIPIENT	idsc:DEFINES_AS

5.2.3 Results

From the assessment of MYDATA and LUCON, we believe that the framework that we propose to identify and assess the main features of DUC solutions is valid to assess to what extent DUC solutions ensure data sovereignty due to three main reasons. First, there

are no features from MYDATA or LUCON that have not been identified in the framework and should be considered. Second, all the features that we set as mandatory for DUC solutions could be assessed for MYDATA and LUCON. Third, the framework has allowed us to identify the main aspects of MYDATA and LUCON and the limitations that they present and should be addressed as future work. They are summed up below.

Regarding policy implementation, several challenges remain. First, while MYDATA enables to define policies using the IDSA UPL through a policy editor, LUCON provides a policy editor to define policies using the LUCON policy language. This makes the agreement of policies between a data provider and a data consumer a difficult task for LUCON if another DUC solution is used in the other side. Furthermore, the LUCON policy language does not provide such expressiveness as the IDSA UPL to define restrictions on the usage of the data. Second, policy quality is not addressed in either MYDATA or LUCON. Thus, policies can be poorly defined and unauthorized data uses or unnecessary policy enforcement may arise. This put the adoption of these DUC solutions into risk. Third, the policy negotiation process is a very new field of research, and although some approaches have been proposed in the context of MYDATA, none have been developed so far. While MYDATA enables data consumers to accept or deny policies from data providers, LUCON propagates the policies from data providers to data consumers without considering the needs of the latter. Finally, regarding the transformation from reference to proprietary policy languages, LUCON has no policy transformation service. However, MYDATA does support it but with a limited expressiveness that needs to be extended to fully support the IDSA UPL.

Regarding policy enforcement, the different approaches used in each DUC solution means that MYDATA and LUCON have different challenges. In terms of the enforcement of permissions and prohibitions, MYDATA provides a wider scope and scalability, as it can support the evaluation of more conditions by improving the scope of the PDP and the PIP. However, LUCON is highly oriented to eval-

uate the labels attached to the data, so the evaluation of conditions is limited to the data flow tracking information. Thus, how other context related information can be retrieved must be studied. In addition, through the PIP and the PXP, MYDATA can self-manage the execution of duties. However, in LUCON duties, with the exception of logging and notification, are delegated to the services involved in data usage. Therefore, MYDATA provides a wider scope for duty enforcement. The approach of LUCON must be studied because the execution of duties by external services highly depends on a previous certification of the capabilities of the services. Finally, both solutions support decision continuity and duty handling. While MYDATA manages these features through the PEP, intercepting usage requests and events that trigger the enforcement of duties, LUCON relies on a routing engine to trigger the enforcement of flow rules each time a data enters or leaves a service and delegates duty handling to the services themselves. The approach followed by LUCON to handle duties, as previously mentioned, should be studied as services should be trusted.

5.3 Experimental Assessment of CAPA for the Wind Energy Domain

Chapter 4 presents the approach to analyse the policy quality for DUC. This section analyses if this approach, which is implemented in the CAPA algorithm, correctly analyses the consistency and minimality policy quality requirements for a set of context-aware DUC policies that support the heterogeneity of conditions. To this end, an experimental assessment of CAPA is performed for a real use case. In particular, we have implemented CAPA in a DUC solution that has been deployed for a wind energy use case. Moreover, because there is a lack of large-scale sets of rules related to DUC, we have created different rule sets. CAPA analyses these rule sets in the experiments performed.

From the results obtained of the experimental assessment, we examined three aspects. First, if CAPA correctly analyses the policy quality. That is, whether CAPA detects all the inconsistencies and redundancies from the rule sets. For this purpose, we have intentionally generated inconsistencies and redundancies in each of the rule sets. Second, we measure the performance improvement that CAPA introduces in the policy analysis procedure by including our contributions related to the management of the heterogeneity of conditions described in Section 4.5. To this end, it is worth mentioning that policy quality has not yet been addressed for DUC policies. As a result, there are no policy analysis approaches for DUC to compare with. Therefore, we have developed and also tested a Basic Policy Analysis (BPA) algorithm. BPA implements the algorithms described in Section 4.7 but omits the contributions presented in Section 4.5 related to the efficient search and comparison of DUC relevant rules. In this regard, conditions are not structured in the PT based on their dependent relationship and they are stored in the same format in which they are expressed. Third, beyond the performance improvement, whether the overall performance provided by CAPA is enough for a real operation scenario is analysed.

On this basis, this section presents the most significant results obtained from the tests carried out on CAPA and BPA in a wind energy use case. To do so, the components of the use case that are most relevant for the implementation of CAPA and BPA are explained. Next, due to the lack of large-scale sets of rules related to DUC, specific rule sets are created. These are detailed. The settings used in the environment where CAPA and BPA are deployed are also detailed. Finally, the results of the experimental assessment are analysed.

5.3.1 Wind Energy Use Case

The wind farm competitiveness is closely connected with the maintenance of wind turbines. Therefore, learning from the data collected from the wind turbine operation is of the utmost impor-

tance (López de Calle, Ferreiro, Roldán-Paraponiaris, & Ulazia, 2019). Due to current reluctance to share data, the data collected from the wind turbine operation is always retained by wind farm operators and OEMs. As the data owners, they are the only actors who usually extract added value from the data (Kusiak, 2016). Thus, other stakeholders in the value chain such as the component suppliers are missing the opportunity to learn from this data. This issue has a major impact not only on the competitiveness of each individual stakeholder, but also on that of the entire value chain. To increase competitiveness, reluctance to share data must be overcome by encouraging trusted data sharing. In this regard, there are already architecture models such as the IDSA RAM which provide a reliable reference for driving the development of distributed architectures that can boost data sharing by providing interoperability, trust and data sovereignty.

Figure 5.3 presents the wind energy use case where different components developed following the IDSA RAM are adopted to enable interoperable, trusted and sovereign sharing of gearbox-related data across the value chain. This use case features a company that deploys sensors which monitor the gearbox of a wind turbine and provide condition data from the lubrication system. From that data, edge computing services developed by a service company estimate gearbox health status. The gearbox is one of the components of a wind turbine that has greatest impact on maintenance. Therefore, sharing the information from those gearbox monitoring sensors with other stakeholders in the value chain such as the component supplier is of great interest because their combined data and expertise can significantly improve gearbox operation and maintenance (e.g., by improving component design). To promote data sharing by providing data sovereignty, a technology provider deploys two IDSA DataSpace Connectors to share data between a data provider and consumer, an Identity Provider to provide trust between the IDSA DataSpace Connectors, and a Vocabulary Provider for resource description. The Connectors and the Vocabulary Provider are directly involved in this DUC demonstration, so they are described in more detail below.

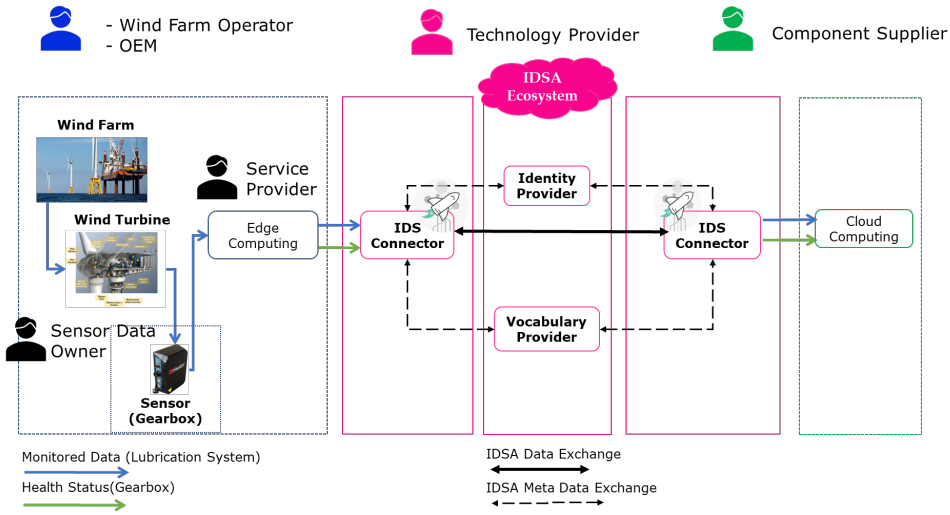

Figure 5.3: IDSA wind energy use case.

IDSA DataSpace Connectors[1] are the technical components responsible for correct data sharing between a data provider (e.g., the wind farm operator) and a data consumer (e.g., a component supplier). Built on the trust provided by the Identity Provider, they are responsible for providing data sovereignty through DUC. To provide interoperability between different IDSA Connectors, each IDSA Connector implements the IDSA Information Model (IM) through a Java Library. Therefore, DUC policies are implemented following the IDSA UPL. In the IDSA UPL, usage restrictions are defined through IDSA contracts (called simply contracts hereafter). A contract comprises metadata (e.g., the date of issue and the participants) and the IDSA Usage Control Policy that, following the IDSA UPL, comprises IDSA Rules that describe the usage control statements that must be examined for policy quality. The policy analysis takes place through the implementation of CAPA or BPA within the IDSA DataSpace Connector in the role of the data provider to analyse offered policies and in the role of the data consumer to analyse requested policies. In the experiment performed CAPA and BPA are only implemented on one side. In particular, in the DUC solution of the data provider.

[1]https://international-data-spaces-association.github.io/DataspaceConnector/

The **Vocabulary Provider**[2] manages and provides a domain-specific Wind Farm Ontology (in this case WFOnt[3]) that describes the resources shared by the IDSA DataSpace Connectors through the IDSA ecosystem. As a result, inconsistencies and redundancies can be detected not only for common, but also, for dependent resources through policy analysis by CAPA and BPA. WFOnt is inspired by the SANDIA Report and reuses the AffectedBy and EEP (Execution-Executor-Procedure) ontology design patterns to discover sensors or actuators that observe or act on a given quality or feature of interest.

5.3.2 Datasets and Setting

CAPA relies on the tree structure described in Section 4.5 to efficiently analyse policies. Te contributions that we introduce in CAPA to efficiently handle the heterogeneity of conditions are presented in the policy tree. Therefore, to make a more detailed analysis of the performance provided by CAPA with respect to the BPA not only for common, but also, for dependent resources we focus on the analysis of the rules defined only for two dependent resources. On this basis, we consider of interest to vary the number of subjects, actions and conditions controlled to analyse their impact on the performance of both CAPA and BPA. In turn, increasing the number of rules to increase the number of subjects, actions and conditions controlled. The number of condition application domains is considered to keep a constant to compare the contributions of CAPA with respect to BPA. Furthermore, inconsistencies and redundancies are intentionally generated to assess if CAPA detects all of them.

Following this approach, we have developed a dataset generator that creates a group of rule sets that are summarized in Table 5.13. These rule sets have been created following a set of guidelines. They are detailed below.

[2]https://github.com/International-Data-Spaces-Association/IDS-VocabularyProvider

[3]https://w3id.org/wfont

Table 5.13: Experimental assessment rule sets.

Rule Set	**Res**	**R**	**AxS**	**C**	L_{ad}
1	2	200	1 x 10	10	5: Time, Connector, Location, Security Level and Count
2			10 x 1		
3			1 x 1	100	
4		2000	1 x 100	10	
5			10 x 10		
6			100 x 1		
7			1 x 10	100	
8			10 x 1		
9		20000	1 x 1000	10	
10			10 x 100		
11			100 x 10		
12			1000 x 1		
13			1 x 100	100	
14			10 x 10		
15			100 x 1		

- Each rule set contains 200, 2000, or 20000 rules.
- For 50 resources available in a use case, which are classified into 5 levels of dependency, the rules of each rule set are equally divided into 10, 1000, or 10000 rules defined for two dependent resources: the deepest one and its parent.
- These 10, 1000 or 1000 rules defined for one resource control the same 1, 10, 100 or 1000 actions performed by the same 1, 10, 100 or 1000 subjects. The product of subjects and actions is referred to as AxS.
- Each action performed by each subject is refined by 10 or 100 conditions distributed in the following 5 application domains: Time, Connector, Location, Security Level, and Count.

CAPA and BPA are implemented in Java 17. All the experiments were performed on a Linux Server running Ubuntu 20.04 with 1 core and 2 GB RAM.

5.3.3 Results

In the following subsections, we first indicate the extent to which CAPA analyse the policy quality by detecting all inconsistencies and redundancies. Next, we focus on the measurement and analysis of the performance provided by CAPA for this purpose compared to BPA. To provide a better insight, CAPA and BPA performance regarding the detection of inconsistencies and redundancies for common and dependent resources is individually analysed below. Whether the performance provided by CAPA can be assumed in a real use case is also analysed.

5.3.3.1 CAPA Policy Quality

As previously indicated, we have intentionally generate inconsistencies and redundancies in all the rule sets. Thus, from the experiments performed on each rule set, the detection capabilities of CAPA can be assessed.

A 100% detection of inconsistencies and redundancies is observed for CAPA. Therefore, we consider that the current policy analysis approach provides the policy quality regarding policy consistency and minimality. As a result, we can state that this policy analysis approach makes improvements in DUC solutions that enable them to boost data sharing in these operational ecosystems by ensuring the data provider with the data sovereignty and optimizing the overall performance on data sharing.

5.3.3.2 Detection of Inconsistencies and Redundancies for Common Resources

As described in Section 4.7.2, the processes that get the condition application domain and reference format from the rule and the one that puts a rule to the PT just get or put a value in a hashmap. Thus, they take *O(1)*. However, the process that finds a *resourceNode* in

the RT and the one that detects inconsistencies and redundancies for common resources become more complex. While CAPA and BPA provide the same performance for the former, for the latter they do not due to the contributions we have introduced in the policy tree. The performance provided by CAPA and BPA for both processes are analysed below.

Find resourceNode

Section 4.7.2.1 theoretically defines that the time complexity to find the resource defined in a rule within the RT is O(1) in the best case if the rule is defined for the root *resourceNode* and $O(Res)$ in the worst case if all the branches of the RT are analysed until the *resourceNode* is found. In the experiment performed, for the rules defined for the deepest resource the time required to find the *resourceNode* is not significant: less than 1 ms for CAPA and BPA.

Detect Inconsistencies and Redundancies

From the *resourceNode* found by the aforementioned process, the time required to detect inconsistencies and redundancies for common resources for a rule depends on the time required to search and compare relevant rules through the corresponding PT. The policy analysis procedure include in CAPA with respect to BPA aims to improve these searches and comparisons.

The experimental assessment proves that regardless of the number of subjects and actions, if the product AxS is the same (e.g., 1×100, 10×10, or 100×1) then the performance provided by CAPA and BPA depends only on the number of conditions (10 or 100). For a clearer analysis, Figure 5.4 and Figure 5.5 represents the total time that it takes in CAPA and BPA to detect inconsistencies and redundancies in all the rules defined for the deepest resource in each rule set based on AxS. In both figures, the volume handled by the policy analysis algorithms increases from 100 to 1000 and 10000 rules. In case of Figure 5.4, AxS ranges from 10 to 100 and 1000 with 10 conditions set in each controlled action for each subject. In Figure 5.5 AxS ranges from 1 to 10 and 100 AxS with 100 conditions.

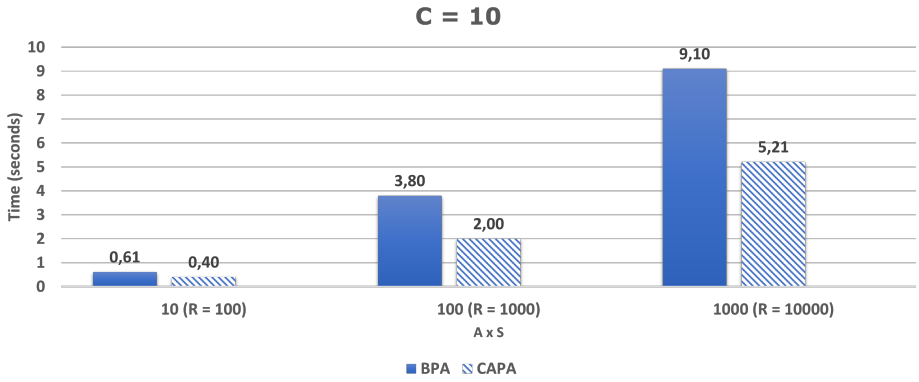

Figure 5.4: Total time required to detect inconsistencies and redundancies for common resources in CAPA and BPA for 10 conditions.

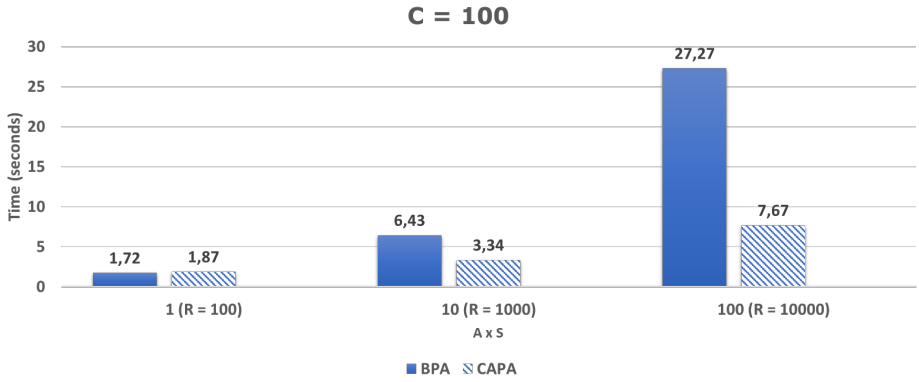

Figure 5.5: Total time required to detect inconsistencies and redundancies for common resources in CAPA and BPA for 100 conditions.

First, the number of conditions that refine each action for each subject (10 or 100) is demonstrated to be the factor with greatest impact on the performance provided by both policy analysis algorithms. For the same number of rules, for example, 1000, if conditions increase from 10 to 100 at the cost of reducing AxS from 100 to 10, i.e., we concentrate conditions in fewer actions and subjects controlled, both algorithms show a loss of performance, but CAPA drops by only 50% while BPA drops by 100%. This issue is also more evident as AxS increases. For 10000 rules CAPA requires 2 more

seconds for 100 AxS and 100 conditions with respect to 1000 AxS and 10 conditions, while in BPA the performance becomes clearly worse, requiring 18 more seconds. Therefore, CAPA proves to be more adaptive to an increasing number of conditions.

Second, for the same AxS, for example, 10, if the number of conditions that refine each action for each subject increase from 10 to 100, it is clear that the performance gets worse as the number of rules increase from 100 to 1000. However, it requires 6 s more in BPA, while in CAPA it requires just 3. Furthermore, it is observed that this issue becomes even more evident as AxS increases. For 100 AxS CAPA requires 5 more seconds for 100 conditions with respect to 10 conditions and in BPA the performance becomes clearly worse, requiring 24 more seconds. Thus, evidencing that CAPA provides a better performance as the number of conditions increase.

Third, the idea of CAPA is to analyse stored rules on demand and afterwards analyse new rules that would like to be added in real time. The time that really affects the operational environment is the time required for the second analysis. In all cases, CAPA takes on average less than 1 ms to analyse a rule. We consider this time to be optimal for an operational environment.

5.3.3.3 Detection of Inconsistencies and Redundancies for Dependent Resources

This process performs a sequential search over the actions for each subject that are controlled for two dependent resources. So, if the product AxS is the same, for the same number of conditions then the performance provided for CAPA and CAPA is observed to be the same. Therefore, to make a more comprehensive analysis, Figure 5.6 and Figura 5.7 represent the total time that it requires for CAPA and BPA to detect inconsistencies and redundancies between the deepest resource and its parent based on AxS for 10 conditions and 100 conditions.

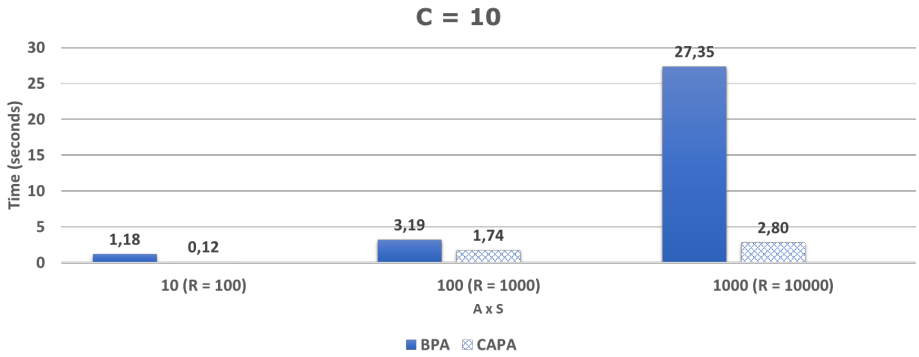

Figure 5.6: Total time required to detect inconsistencies and redundancies for dependent resources in CAPA and BPA for 10 conditions.

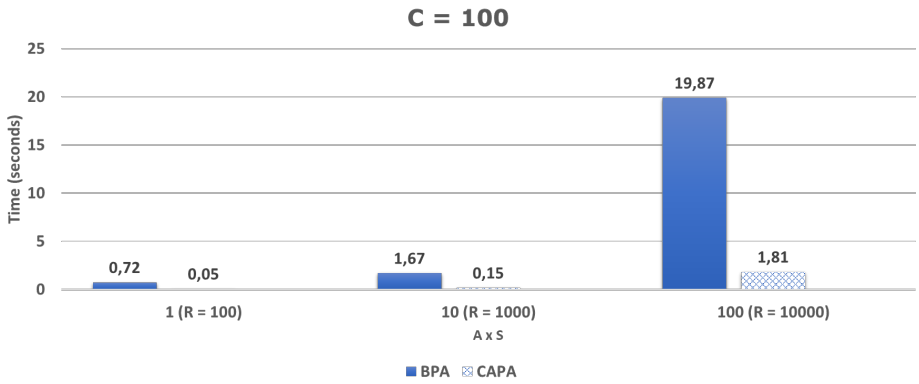

Figure 5.7: Total time required to detect inconsistencies and redundancies for dependent resources in CAPA and BPA for 100 conditions.

First, the product AxS is more critical rather than the number of conditions that refine each action for each subject. For the same number of rules, for example, 1000, it should be noted that reducing AxS from 100 to 10 and increasing the number of conditions that refine each action for each subject from 10 to 100 provides better performance. This is because of the time required for the sequential search for relevant rules for each subject in each action between dependent resources.

Second, CAPA again proves to be more adaptable to an increasing number of conditions. On the one hand, for the same number of conditions, e.g., 10, if AxS increase from 10 to 100, CAPA only takes 1 more second, while BPA 2. However, as AxS increases, i.e, the number of sequential searches increases, the impact of conditions on CAPA and BPA is accentuated. While CAPA only takes 2 more seconds for 10 conditions when AxS increases from 10 to 1000, in BPA it gets 26 more seconds. On the other hand, for the same number of AxS (e.g., 10 AxS), although the conditions increase from 10 to 100 CAPA shows little impact on performance, but BPA takes 500 ms longer. As the number of AxS increases, performance becomes even more critical. For 100 AxS CAPA also shows the same performance for 10 and 100 Conditions, but in BPA there is a drop of 16 s.

Therefore, it is truth that the product of AxS seems to be more critical for the detection of inconsistencies and redundancies for dependent resources with respect to the number of conditions. Nevertheless, due to the contributions introduced in the policy tree for CAPA, it always provides better performance compares to BPA as well as reduces the impact that increasing the numbers of conditions has.

On this basis, as already mentioned before, the idea of CAPA is to analyse stored rules on demand and afterwards analyse each rule that would like to be added in real time. Each time a new rule is about to be analysed, there is no need to perform a complete backtracking process. Instead, resulting permissions from the subject, action and the condition application domain that applies to the rule are recalculated and compared with those from dependent resources. In the experiment performed, to analyse dependent resource conflicts for 1 subject, 1 action and 5 condition application domains in the worst case in which 100 conditions are defined, CAPA takes 50 ms. When a new rule is about to be added, only 1 subject, 1 action and 1 condition application domain is analysed, the ones that refers to the rule. Therefore, from the results of the previous example, we can deduce that it would take 5 times less time, i.e. 10 ms. We consider this time adequate for an operation environment.

5.4 Conclusions

In this chapter, we have analysed the validity of the framework for the identification and assessment of main features for DUC solutions presented in Chapter 3 and the context-aware policy analysis algorithm for DUC presented in Chapter 3.

To validate the framework that we have proposed in Chapter 3, we have examined whether it is valid to know to what extent DUC solutions ensure data sovereignty. For this purpose, we have applied the framework to the most widespread DUC solutions that, to the best of our knowledge, are MYDATA and LUCON.

From the application of the framework, three aspects have been analysed. First, if there are any features of the DUC solutions that have not be taking into account. Second, whether all the features included in the framework can be assessed for the DUC solutions. Third if the strengths and limitations of the DUC solutions can be identified.

From the assessment of MYDATA and LUCON, we realize that the framework we have proposed is valid to assess to what extent DUC solutions ensure data sovereignty. There is no feature from MYDATA and LUCON that have not been identified in the framework. Moreover, all the features identified in the framework have been assessed for MYDATA and LUCON. Furthermore, the main aspects of MYDATA and LUCON as well as their current weaknesses have been identified.

To validate the CAPA algorithm that we have presented in Chapter 3, we have studied whether it correctly analyses the consistency and minimality policy quality requirements for a set of context-aware DUC policies that support the heterogeneity of conditions. To this end, we have implemented CAPA in a DUC solution that has been deployed in a wind energy use case. Moreover, we have created different rule sets. CAPA analyses these rule sets.

From the results obtained of the experimental assessment, we have examined the following three aspects. First, if CAPA detects all inconsistencies and redundancies. To this end, we have intentionally generated inconsistencies and redundancies in each of the rule sets. Second, whether CAPA provides a performance improvement in the policy analysis procedure due to the contributions that our approach introduces to handle the heterogeneity of conditions. In this regard, we have developed and tested a BPA algorithm that implements the algorithms described in Section 4.7 but without the contributions presented in Section 4.5 to efficiently search and compare DUC relevant rules. Third, if the performance provided by CAPA is feasible for a real use case.

With the results obtained from CAPA and BPA after the analysis of all the rule sets, we have validated that CAPA correctly analyses the consistency and minimality policy quality requirements. First, CAPA has reported all the inconsistencies and redundancies that have been generated in the rule sets. This demonstrates that the graph/tree-based modelling approach is a valid approach to analyse the policy quality for DUC policies. Second, CAPA has required less time to analyse all the rule sets with respect to BPA. In turn, the time difference is clearly accentuated as the number of conditions defined for the subjects to perform actions on resources increases. This demonstrates the performance improvement introduced by our contributions in the policy analysis procedure to handle heterogeneous conditions. Third, we consider that the performance provided by CAPA is enough to be implemented in an operational environment. This demonstrates that the graph/tree-based modelling approach is suitable for analysis of the policy quality for DUC policies.

Chapter 6

Conclusions and Further Work

The volume of data generated in the world is growing rapidly. At the same time, data-driven technologies have emerged in the recent years to improve the activities of the organizations. These technologies, require more and more data to remain competitive and cannot just use internal or publicly available data, but also need data from other organizations. In this context, data sharing among independent, unrelated entities becomes increasingly important. Nevertheless, it poses a lot of technological challenges.

A critical issue that prevents a broader data sharing worldwide is the reluctance of organizations to share their data if their self-determination about the usage of their data is not granted. This is referred to as the data sovereignty concern. In the context of data sharing, DUC appears as the key approach to ensure data sovereignty.

This thesis provides new tools to advance the widespread adoption of DUC solutions with the aim of ensuring data sovereignty. In particular, this thesis provides a novel approach on the identification and assessment of the features that DUC solutions must support and the improvement on the policy quality.

First, there exists different DUC solutions already developed. However, as they pursue different objectives, they present different, specific features. We consider that it is very interesting and desirable to identify and characterize the features that DUC solutions must support to ensure data sovereignty, and to construct a framework to assess to what extent existing DUC solutions actually consider them. We think that the identification of the features is important to set the path to ensure data sovereignty through DUC. Also, we believe that the assessment of these features is important to identify the strengths and limitations of DUC solutions. Thus, the most suitable DUC solution for a particular scenario can be selected. Moreover, existing limitations from DUC solutions can be considered as future work.

Second, we identify that policies of low quality in the context of DUC can lead to undesired situations that can compromise data sovereignty and may include additional delays in data sharing. The former jeopardizes the adoption of DUC. The latter may not be assumed in specific scenarios. Therefore, to improve DUC solutions we think that it is important to define an approach to determine if DUC policies are well-defined.

In what follows, the two main contributions of this thesis are presented. They address the issues identified above. Both innovations take as a reference the technological context of DUC. This includes the approaches proposed to implement AC and UC and DUC itself. The main conclusions of this thesis are then presented. Finally, future work is described.

6.1 Contributions

The following subsections outline the main contributions of this thesis. Considering the structure of research of this document, these contributions are presented around two basic pillars that will facilitate the adoption of DUC solutions to advance in sovereign data shar-

ing: The Framework for the Identification and Assessment of Main Features of DUC solutions, and the Context-Aware Policy Analysis Algorithm for Distributed Usage Control.

6.1.1 Framework for the Identification and Assessment of Main Features of Distributed Usage Control Solutions

Chapter 3 presents the framework for the identification and assessment of main features of DUC solutions. This framework identifies the features that DUC solutions must support to meet data sovereignty requirements in the context of data sharing. Furthermore, it enables the assessment of DUC solutions to know to what extent they ensure data sovereignty.

To ensure data sovereignty in the context of data sharing, we consider that two requirements must be met: First, policies defining restrictions on the usage of the data must be implemented considering the needs of the two organizations involved. Second, implemented policies must be enforced from data access to its multiple uses once it has been shared.

Based on a comprehensive analysis of the research related to AC, UC, and DUC, the thesis identifies the features that DUC solutions must support to satisfy these policy implementation and enforcement requirements.

For policy implementation, we define that DUC solutions must support the following features:

- Policy specification: It provides interoperability between offered and requested policies to enable the agreement of policies.
- Policy quality: It avoids unauthorized data uses and longer enforcement times related to poorly defined policies to ensure an adequate performance of DUC.

- Policy negotiation: It resolves conflicts between offered and requested policies and establishes agreed policies to enable data sharing.
- Policy transformation: It implements agreed policies defined using proprietary policy languages of DUC solutions to enable their enforcement.

For policy enforcement, we set that DUC solutions must support the following features:

- Decision continuity: It controls the usage of the data through its life cycle to ensure data sovereignty.
- Permission/prohibition enforcement: It enforces permissions and prohibitions to control usage requests.
- Duty enforcement: It enforces duties to control usage requests.
- Duty handling: It triggers the enforcement of duties to control usage requests.

Considering these features, a framework is proposed that enables the assessment of DUC solutions based on the features they support. This framework provides insight into the scope of DUC solutions by analysing the expressiveness of the policies they manage. In this sense, considering the DUC policy languages proposed so far, we link the assessment framework to what we consider the reference language to define policies in DUC, the IDSA UPL.

As a summary of the framework, the collection of features is identified and for each of them the following questions are answered: What issue the feature addresses? How the feature addresses the issue? How the feature can be assessed?

Below, Table 6.1 summarizes the answers to these questions for each feature with respect to the implementation and policy enforcement requirements established to ensure data sovereignty.

Table 6.1: Main features of DUC Solutions (PS = Policy Specification, PQ = Policy Quality, PN = Policy Negotiation, PT = Policy Transformation, DC = Decision Continuity, P&PE = Permission & Prohibition Enforcement, DE = Duty Enforcement, DH = Duty Handling).

Feature	**What issue it addresses?**	**How it addresses the issue?**	**How it can be assessed?**
Policy Implementation			
PS	Provides interoperability between offered and requested policies to enable the agreement of policies	Uses a reference policy language	Identifying if policies are defined using the IDSA UPL
PQ	Avoids unauthorized data uses and longer policy enforcement times related to poorly defined policies to ensure an adequate operation of DUC	Implements of a policy analysis algorithm	Based on the IDSA UPL, analyzing the expressiveness of the policies that can be managed
PN	Resolves conflicts between offered and requested policies and establishes agreed policies to enable data sharing	Includes a policy negotiation process	Based on the IDSA UPL, analyzing the expressiveness of the policies that can be managed
PT	Implements agreed policies defined using a proprietary language to enable their enforcement	Includes a policy translator	Based on the IDSA UPL, analyzing the expressiveness of the policies that can be managed
Policy Enforcemnet			
DC	Controls the usage of the data from its access to the moment when it has been shared and is used multiple times to ensure data sovereignty	Monitors and intercepts usage requests through a PEP to trigger policy enforcement	Considering if usage requests are monitored and intercepted
P&PE	Enforcement of policies related to permissions and prohibitions to control usage requests	Evaluates conditions through the PDP gathering the information required through the PIP and DMP	Based on the IDSA UPL, analyzing the conditions that can be evaluated
DE	Enforcement of policies related to duties to control usage requests	Performs the actions under the conditions defined through the PXP	Based on the IDSA UPL, analyzing the actions that can be executed
DH	Trigger the enforcement of duties to control usage requests	Monitors and intercepts events that trigger the enforcement of duties through an event handler	Considering the events that can be monitored and intercepted

6.1.2 Context-Aware Policy Analysis Algorithm for Distributed Usage Control

One common limitation identified in all existing DUC solutions is related to the definition of good quality policies. Policies are used to establish the restrictions that govern the usage of the data and then to make decisions on how the data can be used through its life cycle. If policies are not accurate enough, unauthorized data uses, and unnecessary policy enforcement times can appear. In this regard, as indicated previously when setting the main features that must be examined, the policy quality sets the basic requirements that define if policies are correctly defined, so that their enforcement does not lead to these situations that can compromise data sovereignty and put the adoption of DUC solutions at risk or include additional delays on data sharing that may render the adoption of DUC solutions not feasible in an scenario.

This thesis addresses the analysis of the policy quality for DUC policies. To this end, as policy quality has not yet been addressed in the DUC context, we consider the research on policy quality in the AC context as the basis. In this line, it is interesting to point that the quality of AC policies has been largely addressed. For the evolution from existing AC policy analysis approaches to DUC analysis, we identify three key issues to consider. They are described below.

First, DUC policies introduce two features with respect to AC policies that further affect their policy quality. These are the heterogeneity of conditions and the extended control by supporting duties. As aforementioned, the policy quality affects the security and performance of DUC. Although performance is important for DUC, we consider security being more critical, as the latter fully impacts on data sovereignty, while additional delays may be assumed to some extent in certain scenarios. In this context, security is compromised from poorly defined permissions and prohibitions rather than duties. Furthermore, the heterogeneity of conditions makes the appearance of these situations more likely. Therefore, although both features should be considered for the analysis of the policy quality for DUC

policies, this thesis focus the policy quality analysis on the feature that allow DUC policies to support the heterogeneity of conditions.

Second, whereas we consider consistency, minimality, relevance, completeness, and correctness as the policy quality requirements from AC policies that DUC policies should withstand, their impact on DUC policies was examined. Even though all the policy quality requirements may have an impact on the quality of DUC policies, our approach identifies consistency and minimality as the most critical ones. While unauthorized data usage mainly result from inconsistencies, longer policy enforcement times are generally caused by redundancies. To analyse consistency and minimality policy quality requirements, only the policies should be considered. Based on our approach, while completeness and relevance can be analysed considering data usage transactions, correctness can be analysed through the analysis of the intended goals of the policies.

Third, from the approaches proposed for the analysis of the policy quality for AC policies, we consider valid for DUC only those that can detect all the issues that do not satisfy the policy quality requirements. Otherwise, those undesired situations previously identified will appear. Considering this, we think that only the graph-tree based modelling approach is valid for DUC. It is worth mentioning that this approach analyses all the policy quality requirements representing policies as tree or graph structures. As policies are defined as structures composed of different interrelated components, it is not difficult to represent policies as tree or graph structures, where searches among policies can guarantee detecting all the policy quality issues, and can also be quickly performed. Therefore, we consider the graph-tree based modelling approach suitable for DUC.

Considering these three aspects, Chapter 4 presents the approach for the analysis of consistency and minimality policy quality requirements for DUC policies that support the heterogeneity of conditions which is based on the graph/tree-based modelling. From the AC research literature, our approach presents a set of contributions. They are described below.

First, we define a generic structure for DUC policies. Policies defined using any DUC policy language can be mapped to this generic structure. Our approach analyses DUC policies based on this structure. Thus, our approach is valid for any DUC solution.

Second, according to the generic structure, we extend those axioms and definitions assumed in the AC research literature for the analysis of the policy quality for AC policies to deal with the heterogeneity of the conditions that DUC policies present.

Third, based on the definitions and axioms assumed in our approach, we extend initial concerns addressed in the AC research literature for the efficient analysis of AC policies to consider the heterogeneity of conditions. Efficiency refers to the ability to detect any problem affecting the quality of the policies in the shortest possible time. All the inconsistencies and redundancies must be detected to ensure the policy quality. In turn, the performance to do so must be enhanced to reduce its impact on the overall performance of DUC. Considering these concerns, we present a tree structure on which policy analysis can be performed.

Fourth, we propose an assessment method to detect inconsistencies and redundancies between rules based on the methods proposed in the AC research literature and considering the impact that conditions have on the rules.

Following our approach, a policy analysis algorithm denoted as CAPA is presented. Following the graph-tree based modelling approach, CAPA analyses consistency and minimality policy quality requirements for DUC policies that support the heterogeneity of conditions.

6.2 Main Conclusions

We are convinced that the new tools proposed in this thesis advance in sovereign data sharing through DUC solutions. In the short

term, the validation performed and reported in Chapter 5 enables an optimistic vision related to the potential of this thesis results.

The new framework proposed for the identification and assessment of main features to consider in DUC solutions to ensure data sovereignty has been applied to the most widespread DUC solutions. To the best of our knowledge these are MYDATA and LUCON. From the application of the framework, we have analysed three aspects. First whether all the features of the DUC solutions have been considered. Second, if all the features included in the framework can be assessed for the DUC solutions. Third if the strengths and limitations of the DUC solutions can be identified. From the assessment of MYDATA and LUCON, we consider that the framework is valid to assess to what extent DUC solutions ensure data sovereignty. This is because any feature of MYDATA and LUCON has been considered, has been assessed and, as a result, their strengths and weakness regarding data sovereignty have been identified.

Regarding the validity of our approach for the analysis of the policy quality for DUC policies, an experimental CAPA assessment has been performed for a wind energy use case, where we have analysed three aspects: First, If CAPA can detect all inconsistencies and redundancies. Second, the performance improvement that CAPA introduces in the policy analysis procedure by including the management of the heterogeneity of conditions. Third, whether the performance provided by CAPA is enough for a real operation scenario. From the results of the experimental assessment, we have validated that CAPA correctly analyses the consistency and minimality policy quality requirements. That is, CAPA detects all the inconsistencies and redundancies that have been generated in the rule sets. This demonstrates that the graph/tree-based modelling approach is a valid approach to analyse the policy quality for DUC policies. Second, CAPA shows the performance improvement introduced by our contributions in the policy analysis procedure to handle heterogeneous conditions. Third, CAPA provides a suitable performance for an operational environment. This demonstrates that the graph/tree-based modelling approach is suitable for analysis of the policy quality

for DUC policies.

On the medium to long term, it is possible to foresee the way the assessment framework or the CAPA tools will be utilised to enhance existing DUC solutions.

The proposed framework may enable initiatives such as IDSA, working on the definition and design of a distributed infrastructure to enhance the collaboration between organizations through data sharing, to clarify the path to address data sovereignty through DUC solutions. This is of utmost importance since, as discussed throughout this thesis, data sovereignty is the main concern for organizations when it comes to sharing data. Furthermore, the framework may enable the realization of a mature landscape of existing DUC solutions. This can allow service providers to not only to be aware of existing solutions but to be able to select and deploy the most appropriate one based on specific use cases. This landscape may, in turn, enable the identification of weaknesses from specific DUC solutions that should be addressed by the corresponding providers to improve DUC solutions and even general limitations of DUC solutions that should be established as future lines of research to be addressed in the context of DUC.

The CAPA algorithm facilitates software providers with a tool to enhance their developed DUC solutions by ensuring the consistency and minimality policy quality requirements for DUC policies that support the heterogeneity of conditions. In addition, the approach we propose also enables other researchers or the software providers themselves to easily improve the CAPA algorithm. This can be done in two directions. First, by extending the CAPA algorithm to analyse DUC policies that support the extended control by supporting duties. Second, by analysing relevance, completeness, and correctness policy quality requirements. Thus, including CAPA, a DUC solution that will fully ensure the policy quality will be developed.

6.3 Future Work

This thesis has contributed with new insights and results that we expect that will facilitate the adoption of DUC solutions to grant data sovereignty in the context of data sharing. However, there is much research still pending that, although beyond the scope of this thesis, we consider of interest to further increase the quality these solutions. We understand that this thesis might be a first step in an increased structured and engineered approach to the design and development of DUC solutions.

The application of the framework for the assessment of DUC solutions has realised that there are still many limitations that DUC solutions must address to ensure data sovereignty. First, while this thesis addresses the policy quality for DUC policies for the first time, policy negotiation remains as a theoretical proposal and policy transformation has been supported only for the MYDATA proprietary policy language. This results in an incomplete policy implementation approach. Likewise, for policy enforcement, there is still much work remaining to evaluate all the conditions that refine permissions and prohibitions and execute all the duty actions defined within duties. In this regard, we consider that future works will take into account architecture approaches based on ABAC, as the one proposed by MYDATA, as it seems scalable enough as to enforce much more diverse conditions and duties if happens that the reference DUC policy language evolves in terms of expressiveness.

Apart from these major work items, we can also foresee additional work that could improve current thesis results. This is described below.

The framework for the identification and assessment of DUC solutions has been built considering the main requirements that must be satisfied to ensure data sovereignty in the context of data sharing. The future may bring new requirements that have not been identified in this research. Even though such requirements would neither be identified nor supported automatically in the actual framework,

it can easily be extended to incorporate these new requirements including the new features that DUC solutions must support to achieve them.

The framework provides a generic method to assess the expressiveness of the policies each feature of a DUC solution can manage. For example, for the policy quality, the framework enables the assessment of the expressiveness of the policies that can be analysed. In our policy quality analysis approach, this is limited to permission and prohibition rules and specific conditions. For the policy quality, this assessment can be enriched considering the policy quality requirements that are analysed. In our approach, consistency and minimality. Therefore, the framework can be extended to include more refined assessments for each feature that have not been prioritised so far.

In turn, the framework can be extended and enriched with the incorporation of performance assessments for each feature. This may not affect data sovereignty, but it should considered for the applicability of a DUC solution for a use case. A clear example is the time required by our approach to analyse the policy quality for DUC policies, which affects the overall performance of DUC.

Regarding the approach that we propose to analyse the consistency and minimality policy quality requirements for DUC policies that support the heterogeneity of conditions, three aspects should be considered. First, the impact of duties on policy quality should be analysed in depth, and an extension of CAPA should be studied to include adequate support for them. Second, support for completeness, relevance and correctness policy quality requirements should also be studied by complementing CAPA with a transaction-based analysis and considering the intended goals of the policies. Third, the performance that this approach provides to incorporate all this additional work should be examined.

Moreover, it should be considered that CAPA analyses DUC policies on demand. CAPA should evolve towards a real-time analysis of policies. That is, a set of policies that wants to be implemented are

analysed against those that have already been implemented. Thus, providing a real-time insurance that the implemented policies are always of good quality.

References

Aliaksandr, L., Fabio, M., & Paolo, M. (2010). Usage control in computer security: A survey. *Computer Science Review*, *4*(2), 81-99. Retrieved from https://doi.org/10.1016/j.cosrev.2010.02.002

Ammann, P., Lipton, R., & Sandhu, R. (1992). In *[1992] proceedings the computer security foundations workshop v* (p. 148-156).

Aqib, M., & Shaikh, R. (2014). Analysis and comparison of access control policies validation mechanisms. *International Journal of Computer Network and Information Security (IJCNIS)*, *7*(1), 54-69. Retrieved from https://doi.org/10.5815/ijcnis.2015.01.08

Ardagna, C. A., Bussard, L., de Capitani Di Vimercati, S., Neven, G., Paraboschi, S., Pedrini, E., ... Verdicchio, M. (2009). *Primelife policy language.* Retrieved from https://www.w3.org/2009/policy-ws/papers/Trabelisi.pdf (Accessed: 12 September 2022)

Bertino, E., Jabal, A. A., Calo, S., Verma, D., & Williams, C. (2018). The challenge of access control policies quality. *J. Data and Information Quality*, *10*(2), 1-6. Retrieved from https://doi.org/10.1145/3209668

Bettini, C., Jajodia, S., Wang, X., & Wijesekera, D. (2003, 09). Provisions and obligations in policy rule management. *J. Network Syst. Manage.*, *11*, 351-372.

Chung, Ferraiolo, D., Kuhn, D., Schnitzer, A., Sandlin, K., Miller, R., & Scarfone, K. (2019, 2019-02-25). *Guide to attribute based ac-*

cess control (abac) definition and considerations. Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD. Retrieved from https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=927500

Demchenko, Y., de laat, C., & Los, W. (2018, November). Data as Economic Goods: Definitions, Properties, Challenges, Enabling Technologies for Future Data Markets. *ITU Journal: ICT Discoveries, Special Issue 2 "Data for Goods"*. Retrieved from https://doi.org/10.5281/zenodo.2483185

Deng, F., & Zhang, L.-Y. (2017). Elimination of policy conflict to improve the pdp evaluation performance. *Journal of Network and Computer Applications*, *80*, 45-57. Retrieved from https://www.sciencedirect.com/science/article/pii/S1084804516303010

Di Cerbo, F., Some, D. F., Gomez, L., & Trabelsi, S. (2015). Ppl v2.0: Uniform data access and usage control on cloud and mobile. In *2015 ieee/acm 1st international workshop on technical and legal aspects of data privacy and security* (p. 2-7). Florence, Italy: IEEE. Retrieved from https://doi.org/10.1109/TELERISE.2015.9

Di Cerbo, F., Trabelsi, S., Steingruber, T., Dodero, G., & Bezzi, M. (2013). Sticky policies for mobile devices. In *Proceedings of the 18th acm symposium on access control models and technologies* (p. 257–260). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/2462410.2462429

Eitel, A., Jung, C., Brandstadter, R., Hosseinzadeh, A., Bader, S., Kuhnle, C., ... Korth, B. (2021). *Usage Control in the International Data Spaces.* Retrieved from https://internationaldataspaces.org/wp-content/uploads/dlm_uploads/IDSA-Position-Paper-Usage-Control-in-the-IDS-V3..pdf (Accessed: 12 September 2022)

European Comission. (2020). *A European strategy for data.* Retrieved from https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:52020DC0066&from=EN (Accessed: 12 September 2022)

Garrison, W. C., & Lee, A. J. (2015a). Decomposing, comparing, and synthesizing access control expressiveness simulations. In *2015 ieee 28th computer security foundations symposium* (p. 18-32).

Garrison, W. C., & Lee, A. J. (2015b). Decomposing, comparing, and synthesizing access control expressiveness simulations (extended version). *CoRR*, *abs/1504.07948*. Retrieved from http://arxiv.org/abs/1504.07948

Garrison, W. C., Lee, A. J., & Hinrichs, T. L. (2014). An actor-based, application-aware access control evaluation framework. In *Proceedings of the 19th acm symposium on access control models and technologies* (p. 199–210). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/2613087.2613099

Garrison, W. C., Qiao, Y., & Lee, A. J. (2014). On the suitability of dissemination-centric access control systems for group-centric sharing. In *Proceedings of the 4th acm conference on data and application security and privacy* (p. 1–12). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/2557547.2557566

Glennon, M., Kolding, M., Sundbland, M., La Croce, C., Micheletti, G., Raczko, N., ... Osimo, D. (2021). *European DATA Market Study 2021–2023.* Retrieved from https://digital-strategy.ec.europa.eu/en/library/results-new-european-data-market-study-2021-2023 (Accessed: 12 September 2022)

Harrison, M. A., Ruzzo, W. L., & Ullman, J. D. (1976, aug). Protection in operating systems. *Commun. ACM*, *19*(8), 461–471. Retrieved from https://doi.org/10.1145/360303.360333

Hilty, M., Pretschner, A., Basin, D., Schaefer, C., & Walter, T. (2007). A policy language for distributed usage control. In J. Biskup & J. López (Eds.), *Computer security - esorics 2007* (p. 531-546). Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved from https://doi.org/10.1007/978-3-540-74835-9_35

Hinrichs, T. L., Martinoia, D., Garrison, W. C., Lee, A. J., Panebianco, A., & Zuck, L. (2013). Application-sensitive access

control evaluation using parameterized expressiveness. In *2013 ieee 26th computer security foundations symposium* (p. 145-160).

Hosseinzadeh., A., Eitel., A., & Jung., C. (2020). A systematic approach toward extracting technically enforceable policies from data usage control requirements. In *Proceedings of the 6th international conference on information systems security and privacy - icissp,* (p. 397-405). SciTePress.

Hu, V. C., Kuhn, D. R., Ferraiolo, D. F., & Voas, J. (2015). Attribute-based access control. *Computer*, *48*(2), 85-88.

Huang, C., Sun, J., Wang, X., & Si, Y. (2009). Inconsistency management of role base access control policy. In *2009 international conference on e-business and information system security* (p. 1-5).

Iannella, R. (2018). *Open digital rights language (odrl) version 2.2.* Retrieved from `https://www.w3.org/TR/odrl-model/` (Accessed: 12 September 2022)

ITU-T. (1991). *ITU-T X.800 Reccomendation.* Retrieved from `https://www.itu.int/rec/T-REC-X.800-199103-I/en` (Accessed: 12 September 2022)

Jabal, A. A., Davari, M., Bertino, E., Makaya, C., Calo, S., Verma, D., ... Williams, C. (2019). Methods and tools for policy analysis. *ACM Comput. Surv.*, *51*(6), 1-35. Retrieved from `https://doi.org/10.1145/3295749`

Jarke, M., Otto, B., & Ram, S. (2019). Data sovereignty and data space ecosystems. *Bus Inf Syst Eng*, *61*, 549–550. Retrieved from `https://doi.org/10.1007/s12599-019-00614-2`

Jung, C., & Dörr, J. (2022). Data usage control. In B. Otto, M. ten Hompel, & S. Wrobel (Eds.), *Designing data spaces : The ecosystem approach to competitive advantage* (pp. 129–146). Cham: Springer International Publishing. Retrieved from `https://doi.org/10.1007/978-3-030-93975-5_8`

Jung, C., Eitel, A., & Schwarz, R. (2014). Enhancing cloud security with context-aware usage control policies. In E. Plödereder, L. Grunske, E. Schneider, & D. Ull (Eds.), *Informatik 2014* (p. 211-222). Bonn: Gesellschaft für Informatik e.V.

Katt, B., Zhang, X., Breu, R., Hafner, M., & Seifert, J.-P. (2008).

A general obligation model and continuity: Enhanced policy enforcement engine for usage control. In *Proceedings of the 13th acm symposium on access control models and technologies* (p. 123–132). New York, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/1377836.1377856

Kelbert, F., & Pretschner, A. (2015). A fully decentralized data usage control enforcement infrastructure. In T. Malkin, V. Kolesnikov, A. B. Lewko, & M. Polychronakis (Eds.), *Applied cryptography and network security* (pp. 409–430). Cham: Springer International Publishing.

Kelbert, F., & Pretschner, A. (2018, apr). Data usage control for distributed systems. *ACM Trans. Priv. Secur.*, *21*(3). Retrieved from https://doi.org/10.1145/3183342

Kusiak, A. (2016). Renewables: Share data on wind energy. *Nature*, *529*, 19-21.

Li, N., Mitchell, J. C., & Winsborough, W. H. (2005, may). Beyond proof-of-compliance: Security analysis in trust management. *J. ACM*, *52*(3), 474–514. Retrieved from https://doi.org/10.1145/1066100.1066103

López de Calle, K., Ferreiro, S., Roldán-Paraponiaris, C., & Ulazia, A. (2019). A context-aware oil debris-based health indicator for wind turbine gearbox condition monitoring. *Energies*, *12*(17). Retrieved from https://www.mdpi.com/1996-1073/12/17/3373

Munawer, Q., & S, R. S. (1999). Simulation of the augmented typed access matrix model (atam) using roles. In *In proceedings of infosecu99 international conference on information and security.*

Nyanchama, M., & Osborn, S. (1996). Modeling mandatory access control in role-based security systems. In D. L. Spooner, S. A. Demurjian, & J. E. Dobson (Eds.), *Database security ix: Status and prospects* (pp. 129–144). Boston, MA: Springer US. Retrieved from https://doi.org/10.1007/978-0-387-34932-9_9

Osborn, S., Sandhu, R., & Munawer, Q. (2000, may). Configuring role-based access control to enforce mandatory and discre-

tionary access control policies. *ACM Trans. Inf. Syst. Secur.*, *3*(2), 85–106. Retrieved from https://doi.org/10.1145/354876.354878

Otto, B., Steinbuss, S., Teuscher, A., & Lohmann, S. (2020). *IDS Reference Architecture Model.* Retrieved from https://internationaldataspaces.org/wp-content/uploads/IDS-Reference-Architecture-Model-3.0-2019.pdf (Accessed: 11 October 2022)

Park, J., & Sandhu, R. (2004). The uconabc usage control model. *ACM Trans. Inf. Syst. Secur.*, *7*(1), 128–174. Retrieved from https://doi.org/10.1145/984334.984339

Park, J., Zhang, X., & Sandhu, R. (2004). Attribute mutability in usage control. In C. Farkas & P. Samarati (Eds.), *Research directions in data and applications security xviii* (p. 15-29). Boston, MA: Springer US. Retrieved from https://doi.org/10.1007/1-4020-8128-6_2

Penelova, M. (2021, 12). Access control models. *Cybernetics and Information Technologies*, *21*, 77-104.

Pretschner, A., Hilty, M., & Basin, D. (2006). Distributed usage control. *Commun. ACM*, *49*(9), 39–44. Retrieved from https://doi.org/10.1145/1151030.1151053

Pretschner, A., Massacci, F., & Hilty, M. (2007). Usage control in service-oriented architectures. In *Trust and privacy in digital business.*

Reinsel, D., Gantz, J., & Rydning, J. (2018). The digitization of the world from edge to core.

Sandhu, R. (1998). Role-based access control. In M. V. Zelkowitz (Ed.), (Vol. 46, p. 237-286). Elsevier. Retrieved from https://www.sciencedirect.com/science/article/pii/S0065245808602065

Sandhu, R., Coyne, E., Feinstein, H., & Youman, C. (1996). Role-based access control models. *Computer*, *29*(2), 38-47.

Sandhu, R., & Munawer, Q. (1998). How to do discretionary access control using roles. In *Proceedings of the third acm workshop on role-based access control* (p. 47–54). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/286884.286893

Sandhu, R., & Park, J. (2003). Usage control: A vision for next generation access control. In V. Gorodetsky, L. Popyack, & V. Skormin (Eds.), *Computer network security* (p. 17-31). Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved from https://doi.org/10.1007/978-3-540-45215-7_2

Scaria, E., Berghmans, A., Pont, M., Arnaut, C., & Leconte, S. (2018). *Study on data sharing between companies in europe : final report.* Publications Office. Retrieved from https://doi.org/10.2759/354943

Schuette, J., & Brost, G. S. (2018). Lucon: Data flow control for message-based iot systems. In *2018 17th ieee international conference on trust, security and privacy in computing and communications/ 12th ieee international conference on big data science and engineering (trustcom/bigdatase)* (p. 289-299). New York, NY, USA: IEEE. Retrieved from https://doi.org/10.1109/TrustCom/BigDataSE.2018.00052

Sun, L., Wang, H., Tao, X., Zhang, Y., & Yang, J. (2011). Privacy preserving access control policy and algorithms for conflicting problems. In *2011ieee 10th international conference on trust, security and privacy in computing and communications* (p. 250-257).

Trabelsi, S., Sendor, J., & Reinicke, S. (2011). Ppl: Primelife privacy policy engine. In *2011 ieee international symposium on policies for distributed systems and networks* (p. 184-185). Pisa, Italy: IEEE. Retrieved from https://doi.org/10.1109/POLICY.2011.24

Tripunitara, M. V., & Li, N. (2004). Comparing the expressive power of access control models. In *Proceedings of the 11th acm conference on computer and communications security* (p. 62–71). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/1030083.1030093

Tripunitara, M. V., & Li, N. (2007, apr). A theory for comparing the expressive power of access control models. *J. Comput. Secur.*, *15*(2), 231–272.

United States Department of Defense. (1985). *Trusted Computer System Evaluation Criteria.* Retrieved from https://web.archive.org/web/20060527214348/

`http://www.radium.ncsc.mil/tpep/library/rainbow/5200.28-STD.html` (Accessed: 12 September 2022)