
# Compliance Ontology

## Deliverable D3.1

**Editors**

Georgios Lioudakis (ABOVO)
Davide Cascone (BAK)

**Reviewers**
Nikolaos Dellas (SLG)
Marwan Hassani (TUE)

**Date**
28th February 2019

**Classification**
Public

BPR4GDPR has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 787149

## Table of Contents

# 1 Introduction

The overall goal of the Work Package 3 is to provide a comprehensive framework for regulating the BPR4GDPR operations by means of security and privacy policies that will reflect the GDPR and related legislation. To this end, the final outcome of this Work Package will be a rule-based access and usage control framework, along with the necessary mechanisms for knowledge extraction and decision making, as well as the appropriate information ground, that will be able to drive the compliant execution of operations. This translates to two different missions as regards the role of the policy framework. On the one hand, it is meant to provide the means for system governance in real-time, in the sense that it sets the rules that regulate the operation of BPR4GDPR components. On the other hand, policies comprise the knowledge base that feeds the procedure of process re-engineering, towards compliant by design process models.

This Deliverable comprises the first of Work Package 3, and deals with the concept of the, so-called, Compliance Ontology. Simply put, philosophy defines ontology as the *study of being*, focusing on issues and questions such as what things exist, what categories do they belong to, and how they are related to each other. In line with this, the purpose behind the Compliance Ontology is to provide the appropriate formalisms describing all important aspects related with compliance, focusing on fundamental entities, their categories, and their relations. In this context, the Compliance Ontology serves as the ontology of the GDPR universe, or, in more practical terms, a high-level codification of the GDPR into concepts that need to be taken into consideration by the BPR4GDPR policy framework, as well as by the privacy-aware process re-engineering.

In other words, the Compliance Ontology provides the knowledge ground upon which policies will be formalised as sophisticated and fine-grained rules. This ground incorporates a variety of concepts, including the data types under which personal data fall, roles of the entities requesting and processing personal data, operations and services performed over personal data, attributes of all the involved entities, purposes of requesting/processing data, among others. The Compliance Ontology also considers the interrelations among the identified concepts and provides for their thorough semantic organisation, by specifying hierarchies of data types reflecting generalisation/particularisation, the detail level and the inclusion of some data types to another. These hierarchies provide for comprehensive and simpler specification and formalisation of security and data protection rules, at detailed granularity and at different abstraction levels; this will allow for attribute-based data collection and overall handling, and managing of all associated constraints, including retention periods and the application of protection measures.

The Compliance Ontology constitutes the bridge between the legal and the technical work; as a matter of fact, it reflects the work performed by the project Task 2.2 “Regulatory analysis” and the thorough review of the quite complex data protection regulation landscape, providing the intermediate step of the definition of appropriate formalisations that are able to capture the legal concepts, which is fundamental for the specification of the policy framework and specific rules thereof, in the context of the project Task 3.2 “Rule-based access and usage control”.

Therefore, having the regulatory analysis as the starting point, the next Section identifies the challenges that the compliance framework should address. In other words, it summarises, from a technical point of view, the main issues that BPR4GDPR has to face in order to achieve its goals towards providing solutions for compliant systems and operations. These challenges explicitly mention various concepts that should be considered in the

framework, whereas they implicitly generate various ontological aspects as regards the *being* of entities related to compliance, and their relations.

Based on these findings, Section 3 delves into the core of the Compliance Ontology, being the BPR4GDPR universe of conceptual entities, referred to as the Information Model and its semantic implementation, the Information Model Ontology (IMO). The description covers the basic concepts, the model in terms of entities and relations, the IMO, as well as the default population of the ontology. The latter comprises an indicative identification of the typical instances that are more or less present —or possible— in all organisations that process personal data, regardless of type, size and sector. Its aim is to comprise the default minimum configuration of BPR4GDPR solutions.

Going a step further, the universe defined by the Compliance Ontology is characterised by certain *laws* that regulate the underlying entities. Intuitively, these laws are the rules stemming from the GDPR. Whereas the rule-based framework is not the scope of the Compliance Ontology —nor of this Deliverable— it was deemed appropriate to set the baseline rules already at this time, anticipating the detailed documentation in the context of the Deliverable D3.2 “Initial specification and prototyping of the policy framework”. To this end, a list of fundamental rules is provided in Section 4, along with a short overview of the associated formalisms centred around the cornerstone concepts of actions and rules.

# 2 From regulatory requirements to technical aspects

## 2.1 The Regulation in a nutshell

Following the analysis performed in project Task 2.2 “Regulatory Analysis” with special focus on the GDPR, along with associated regulatory acts and guidelines, various requirements that should be satisfied by the BPR4GDPR solutions have emerged, providing the ground towards the development of mechanisms that will facilitate compliance. The requirements are summarised in the following.

RR-1. **Lawfulness of the data processing**: The system should be able to examine whether the data processing complies with applicable laws and regulations.

RR-2. **Purposes for which data are processed**: The system should provide the means for identifying the data processing purposes, which must be lawful and made explicit to the data subject. Moreover, they should be able to check these purposes to avoid that data processed for a specific purpose may be further processed for purposes that are incompatible with these for which data have been collected.

RR-3. **Necessity, adequacy and proportionality of the data processed**: The system should be able to guarantee that only the data that are functional, necessary, relevant, proportionate and not excessive with regard to the sought processing purpose are processed.

RR-4. **Quality of the data processed**: The system should provide that the data processed are correct, accurate and updated. Inaccurate data must be deleted or rectified; outdated data must be deleted or updated.

RR-5. **Identifiable data**: The system should provide the means for keeping the data processed in identifiable form only for the time necessary to achieve the sought processing purpose.

RR-6. **Notification and other authorisations from competent Data Protection Authority**: The system should comply with the notification requirement and with the provisions on the authorisations of competent Data Protection Authority. Moreover, the system should provide for means that allow communications between the system and the competent Data Protection Authority.

RR-7. **Information to the data subjects**: The system should be able to provide a mechanism for informing the data subject that the data are processed according to the GDPR and the applicable local data protection legislation.

RR-8. **Consent and withdrawal of consent**: The system should guarantee that when requested by applicable data protection legislation, the data subject's consent to the data processing is required, and that the data processing is performed according to the preferences expressed by the data subject. Further, withdrawal of consent and an objection to data processing by the data subject should be handled appropriately.

RR-9. **Exercising rights of the data subject**: The system should enable the data subject to exercise the rights acknowledged by applicable data protection legislation in relation to intervention in the data processing (for example the right to access data, to ask for data rectification, erasure, blocking, the right to object the data processing, etc.).

RR-10. **Data security and confidentiality**: The system should be secure in order to guarantee the confidentiality, integrity, and availability of the data processed. Moreover, the system should provide that the listening, tapping,

storage or other kinds of interception or surveillance of communications and the related traffic data may be performed only with the data subject's consent or when allowed by applicable legislation for public interest purposes.

RR-11. **Special categories of data**: The system should be able to guarantee that the processing of special categories of data (such as health data or religious belief[1]) is performed in compliance with the specific requirements that the applicable data protection legislation sets forth for said categories of data.

RR-12. **Access limitation**: The system should provide for an authorisation procedure that entails differentiated levels of access to the data and also for recording the accesses to the data.

RR-13. **Data storage**: The system should be able to automatically delete (or make anonymous) the data when the pursued processing purpose is reached or in case of elapse of the data retention periods specified under applicable legislation.

RR-14. **Dissemination of data to third parties**: When components of the service logic are outsourced to third party providers and, in this context, personal data are disseminated for being processed, the system should be able to provide certain guarantees that the consequent processing of information complies with the underlying fair data practices and the contract with the data subject.

RR-15. **Transfer of data to third countries**: The data dissemination principle described above applies especially when data are transferred to third countries, possibly with essentially different legislation regarding personal data collection and processing. The system should be able to provide for compliance with the specific provisions ruling on transfer of data.

RR-16. **Supervision and sanctions**: The competent Data Protection Authority should be provided with the means for supervising and controlling all actions of personal data collection and processing.

## 2.2 Lessons learned

The elaboration of privacy principles and requirements has been the subject of various studies and extensive research (e.g., [1][2][3]). Rethought from the point of view of BPR4GDPR, aiming at the development of a compliance framework, the corresponding principles and requirements converge to the following challenges that comprise the foundational ground driving the design of BPR4GDPR Compliance Ontology:

*Purpose*: The “purpose principle” is essential for privacy awareness, being a core part of data collection and processing lawfulness; to this end, a compliance framework should provide for purpose specification and binding.

*Access rights enforcement*: Any privacy violation certainly includes illicit access to personal data, resulting in the emergence of the field referred to as privacy-aware access control [4]. Therefore, BPR4GDPR should provide for access and usage control enforcement, whereas, in process terms, access and usage control policies should be embedded in the process models; this implies comprehensive task specification, as well as a holistic, process-wide approach of control.

[1] See art. 9 GDPR.

*Multi-aspect access rights definition*: Security and data protection solutions should incorporate various criteria in access and usage control decisions. Starting from the very fundamental primitive of *which user* holding *which role* is performing *which action* on *which object*, a variety of parameters should be considered, including the purpose for which an action takes place, the context under which an action is going to be executed, *when* and for *how long*, the history of actions that have preceded, etc.

*Privacy-aware information flows*: Beyond controlling access and usage, a compliance model should provide for the specification of acceptable patterns as far as the flow of data is concerned; this implies, for instance, the prevention of some data to be communicated from a system to another, whereas the latter may be per se allowed to receive the same data by a third system.

*Unlinkability*: Along the same line, a compliance model should provide support for preventing linkability. Whereas privacy-aware information flow refers to "direct" passing of data among systems, processes or people, the need for unlinkability reflects a generalisation towards mutually exclusive availability or processing of data, either explicitly or implicitly.

*Separation and Binding of Duty (SoD/BoD)*: Similarly, SoD and BoD constraints should be possible to be specified and enforced, since they hold an important position among authorisation requirements, serving, among others, conflicts avoidance and unlinkability.

*Complementary actions*: In several cases, access to or usage of data should be complemented by certain actions that should follow the collection and/or processing of information. These are often referred to in the literature as "privacy obligations" and may concern, for instance, the application of immediate protection measures, the interaction with the data subjects (e.g., in terms of information or request for consent), and the enforcement of data retention provisions.

*Context-awareness*: It has become apparent that effective security and privacy policies largely depend on contextual parameters. Therefore, the BPR4GDPR compliance framework should incorporate the corresponding aspects, in terms of restrictions over contextual parameters and events, and be enabled to impose different access and usage rights according to the applicable constraints.

*Security mechanisms*: The basis for the protection of resources and in particular of transmitted data is certainly the security of information and communication systems and communication channels. In this context, systems should be secure so as to be able to guarantee the confidentiality, integrity and availability of the data, as well as to prevent any unauthorised interception or monitoring of data, while the appropriate communication protection mechanisms should be in place. Therefore, a compliance-oriented solution should ensure the integration of the necessary security mechanisms in the underlying processes on a case-by-case basis.

*Data subjects' rights*: The GDPR has entrenched the rights of the data subjects, putting emphasis on existing rights and incorporating new ones. Rights such as the rights to access, rectification, objection, restriction and notice, the right to be forgotten and the right to data portability bring a new perspective on existing rights and include new obligations for organisations. The BPR4GDPR compliance framework shall ensure the enforcement of these rights, primarily by adopting the corresponding policies and providing the means for their enforcement.

*By design and by default*: "Privacy by design" as a concept has existed for years now, but it only became explicitly part of a legal requirement with the GDPR, along with the concept of "privacy by default". Under this

requirement, organisations need to design compliant policies, procedures and systems at the outset of any product or process development. In practical terms as regards the BPR4GDPR compliance framework, by design largely implies, along with the need to have compliant systems and procedures, to put in place and enforce rules fostering compliance, as well as to ensure that these rules become integral parts for processes.

*Semantics*: Vertical to all the above is the need for precise semantics of the underlying concepts; data, actors, actions, context, purposes, events, among others, should be semantically defined, fostering transparency, accountability and effectiveness in terms of privacy. This allows also for the classification of data as regards their sensitivity and speciality, and enables the fine-grained definition of all underlying concepts, therefore fostering the specification of fine-grained and comprehensive handling policies.

# 3 Information Model

This section focuses on the main pillar of the Compliance Ontology, the BPR4GDPR Information Model, that reflects the entities that participate in the lifecycle of processes and, in general terms, the typical operational patterns of organisations that are subject to the project scope.

## 3.1 Basic concepts

The day-to-day operation of an organisation involves a variety of entities, like machines, users and data. BPR4GDPR considers two representation levels (Figure 1); the *concrete level* refers to well-specified entities, e.g., named humans, while the *abstract level* enables referring to entities by using abstractions, especially their semantic type and attributes. This Information Model provides the ground for the specification of the policies regulating the system's operation, as well as the process models providing the blueprint for formalising the operations taking place within the organisation.

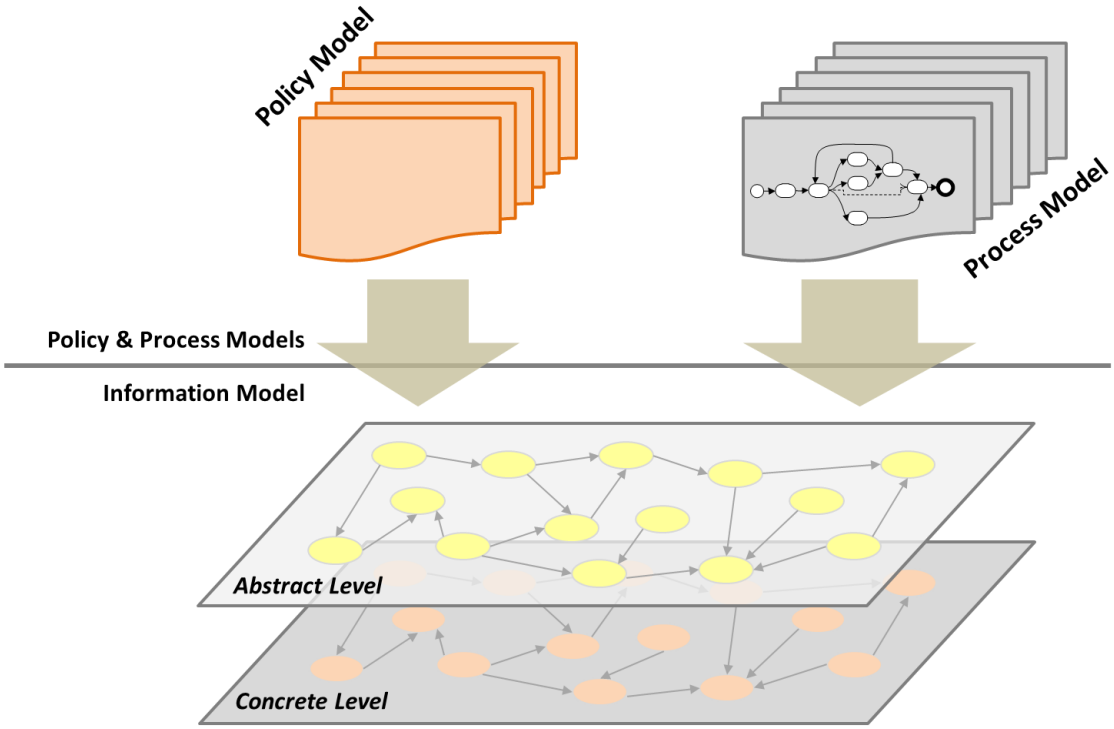


**Figure 1: BPR4GDPR foundational models**

At a concrete level, the set of *Users (U)* represents human entities, while this of *Organisations (Org)* describes internal divisions (e.g., departments) or external parties (e.g., sub-contractors). The various machinery comprise the *Machines (M)* set, providing hosting to *Operation Containers (OpC)* that offer *Operation Instances (OpI)*. The latter correspond to actual implementations of functionalities, while Operation Containers bundle collections of Operation Instances provided by the same functional unit[2]. Finally, information comprises the set of *Data (D)*, whereas *Events (E)* take place and may lead to actions for responding thereof.

[2] To make this more clear, using Web Services terms, Operation Containers correspond to a service *interface*, whereas Operation Instances represent the associated *operations*.

All above elements constitute instantiations of their semantic equivalents described at the abstract level. Users are assigned with *Roles (R)*, Operation Instances provide implementations of *Operations (Op)*, while data, organisations, machines, operation containers and events have types, reflecting the semantic class they fall under; thus, sets of *Data Types (DT)*, *Organisation Types (OrgT)*, *Machine Types (MT)*, *Operation Container Types (OpCT)* and *Event Types (ET)* are defined. The semantic model also includes *Context Types (ConT)*, enabling the definition of contextual parameters, *Attributes (Att)*, leveraged for describing properties and characteristics of other elements, and *Purposes (Pu)* justifying access requests, as well as any other type of action that takes place during the system operation.

| Abstract Level | Concrete Level | Description |
|---|---|---|
| Data Types (*DT*) | Data (*D*) | Data being collected and/or processed, organised according to their semantic types |
| Roles (*R*) | Users (*U*) | Human users assigned with roles reflecting their responsibilities inside an organisation |
| Operations (*Op*) | Operation Instances (*OpI*) | Operations reflect all actions that can take place in the context of the system's operation |
| Operation Container Types (*OpCT*) | Operation Containers (*OpC*) | Components or other functional structures that typically offer a set of operations together |
| Machine Types (*MT*) | Machines (*M*) | Hardware (in the typical case) components hosting operation containers |
| Organisation Types (*OrgT*) | Organisations (*Org*) | The various domains within which actions are performed |
| Event Types (*ET*) | Events (*E*) | Expected or unexpected events that may affect the operation of an organisation, or may call for actions in response |
| Context Types (*ConT*) | Context keys and values | Real-time parameters that should be considered in decision making, such as spatial, temporal, environmental values |
| Purposes (*Pu*) | (no concrete representation) | Purposes for which actions take place, processes are executed, and access to resources is requested |
| Attributes (*Att*) | Attribute keys and values | Characteristics further describing members of the other sets |

**Table 1: Concepts of the Information Model**

While most of these notions are either typically present in state-of-the-art models or intuitively self-explained, a few remarks are deemed necessary for some of them. Specifically, the *OpC* and *OpCT* are introduced in order to model components or other functional structures that typically offer a set of operations together. For instance, an `ePrescritpionSystem` clusters several operations related with the management of an electronic medical prescription, such as `CreatePrescription` and `ViewPrescription`. Apart from the convenience it introduces regarding several modelling aspects (such as the inheritance of attributes), these structures are also helpful for describing a variety of concepts related with ''horizontal'' dependencies and transfer of characteristics. Moreover, the machines play a fundamental role in digital systems operation and, therefore, the BPR4GDPR models cannot be limited to a level of abstraction exclusively centred around functionalities; in any case, functionalities are provided by machines which, on the one hand, are characterised by attributes (e.g., topological ones) that may be inherited to the hosted functionalities and, on the other hand, create inherent dependencies between the hosted functionalities. Finally, organisations are explicitly modelled at both the abstract and concrete levels, because there has been considered the case that, within a unified model, similar rules may be differently defined for heterogeneous organisation types. For instance, within an

organisation of type `BillingServiceProvider`, authorisations regarding the access to customers' data will not be the same with the ones defined for the same data within a `TelecomOperator`, being the Data Controller entity that assigns billing data processing to the `BillingServiceProvider`. Even more, an organisation that participates in cross-domain collaborations will likely define different authorisations for different types of third-party organisations based on their type.

All concepts summarised in Table 1 comprise graphs of elements that are characterised by relations; the latter are implemented by predicates defining AND- and OR-hierarchies and enabling the inheritance of attributes and rules, as well as the specification of dependencies. For instance, and with respect to the *DT* graph, three partial order relations are defined: *isA*($dt_i$, $dt_j$), *moreDetailedThan*($dt_i$, $dt_j$) and *isPartOf*($dt_i$, $dt_j$), where $dt_i$, $dt_j \in DT$, reflecting the particularisation of a concept, the detail level and the inclusion of some data types to another, respectively. Moreover, the model specifies the necessary predicates in order to link concepts from different graphs; for example, the predicate *mayActForPurposes*($r$, $\langle pu \rangle^k$), where $r \in R$, $\langle pu \rangle^k \subseteq \mathcal{P}(Pu)$, indicates the legitimate purposes $\langle pu \rangle^k$ for which the users assigned with the role $r$ may act. The relations forming hierarchies among the members of a set, as well as the cross-concept relations, are presented in Figure 2.

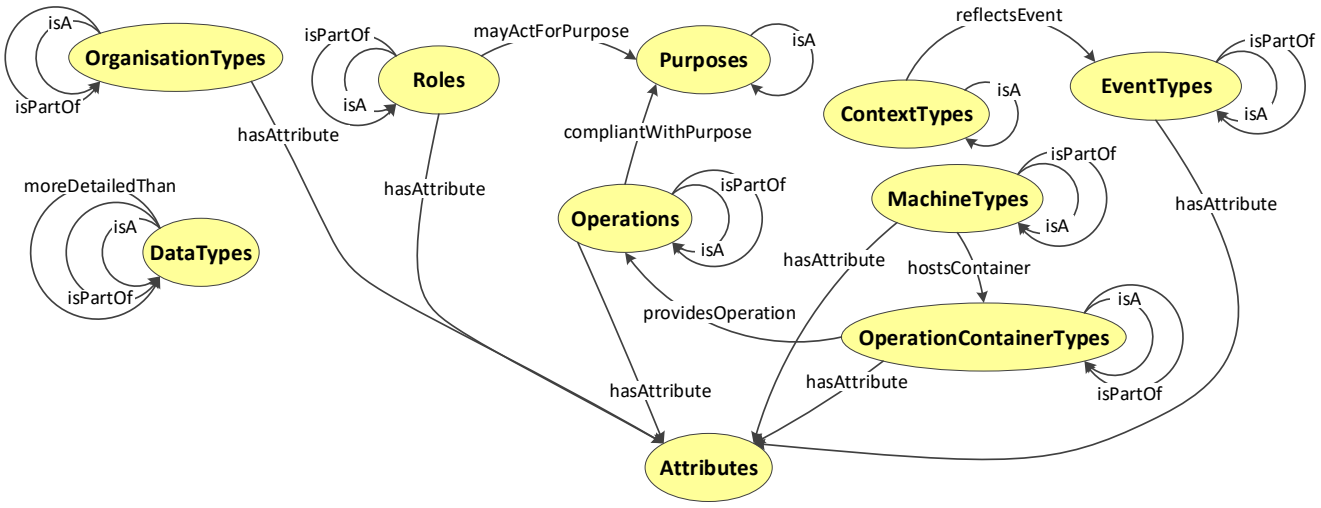


**Figure 2: Information Model basic entities**

In the following, some of the presented concepts are described in more detail, in order to highlight the core aspects of the BPR4GDPR approach related to compliance-awareness.

## 3.2 Core entities of the Information Model

This Section put the focus on the basic elements of the Information Model —data and types thereof, users and their roles, operations and the structures being their containers, events and contextual parameters, purpose and attributes—, delving into some details regarding the concepts themselves, the internal organisation, as well as the external relations, notably the relations among concepts of different types.

### 3.2.1 Data and their types

Data comprise a core concept within any organisation, at least to the extent they pertain to its daily operation. Data are characterised by a data type, that is, the semantic class they fall to. In fact, such semantics of the data items play a critical role on how the data should be processed, as well as on the enforcement of provisions

related with privacy. In that respect and as afore mentioned, two sets were defined; *Data* (*D*), representing the data at a concrete level, and *DataTypes* (*DT*), reflecting the semantic types of data at an abstract level.

The data types are organised by means of three relations, notably *isA*($dt_i$, $dt_j$), *moreDetailedThan*($dt_i$, $dt_j$) and *isPartOf*($dt_i$, $dt_j$), that define transitive and anti-symmetric partial orders of the data types and reflect, respectively, the particularisation of a concept, the detail level, and the inclusion of a data type to another. Regarding the *moreDetailedThan*($dt_i$, $dt_j$) relation, it allows for effectively tuning the accuracy of disclosed data, in order to meet the so named ''proportionality principle'', which requires that the personal and business data may be gathered and processed only to the extent that they are adequate, relevant and not excessive if compared with the monitoring function for which they are collected by the system. For instance, the following apply:

- *isA*(`SocialSecurityNumber`, `Identifier`)- the particularisation of a concept
- *isPartOf*(`LastName`, `FullName`)- the inclusion of a data type to another
- *moreDetailedThan*(`YearOfBirth`, `isAdult`)- the detail level

Regarding inheritance of attributes, each particularised data type has all the characteristics of the generic data type, plus additional characteristics that make it special. For instance, a data type may be complemented by an attribute indicating its storage period; this attribute is inherited to all the particularised data types. Inheritance of attributes can be inferred also for data types interrelated through *isPartOf* and *moreDetailedThan* relations, depending though on the nature of the corresponding attribute.

### 3.2.2 Users and Roles

Evidently, humans constitute a pervasive link in the operational chain of organisations, and data collection and processing activities thereof; regardless the degree of automation of a procedure, there is always some human that designs, initiates, executes a process and/or is in charge of the underlying software and/or hardware components. On the other hand, users are assigned with roles, reflecting, e.g., their responsibilities inside an organisation, through the following predicate, meaning that the user *u* is assigned with the roles $\langle r \rangle^k$:

- *assignedWithRoles*($u$, $\langle r \rangle^k$), where $u \in U$ and $\langle r \rangle^k \subseteq \mathcal{P}(R)$.

Roles are also organised by means of the *isA* and *isPartOf* relations, reflecting particularisation and membership to a complex role, respectively; the latter models cases such as the participation of a role `ManagingDirector` in the `BoardOfDirectors`. Moreover, roles, apart from explicitly defined attributes, acquire all the attributes of their ancestors in the role hierarchies; the `Employee` role may bear a `Schedule` attribute, which is consequently inherited to the `Accountant` role as the latter relates to the former role through the *isA* predicate. In the case of *isPartOf* relation, some attributes of the composite role can be inferred from the member roles' corresponding attributes.

### 3.2.3 Operations and containers

Operations reflect all actions that can take place in the context of the system's operation, constituting the "heart" of the *Action* structure as it will be described in Section 4.1. There exist also cases where operations are treated as subjects of actions, especially within highly automated systems, where operations trigger the execution of others.

Operations comprise a set of semantically defined elements (*Operations* (*Op*) set) and are characterised by different granularities; starting from very fundamental processing units that can be characterised as *atomic*, functional components can be organised in different structures and create complex compositions that can go up to a very high level and represent generic tasks, thus providing convenient abstractions of the underlying complex compositions. In other words, operations form hierarchies that reflect the different levels of granularity, based on the semantic definitions of the operations themselves. The *isA* relation reflects an OR structure indicating different alternative operations that implement the same operation, while the *isPartOf* one comprises an AND relation, requiring all the low-level operations to take place in order for the high-level operation to be fulfilled.

For example, the following state, respectively, that `AuthenticateUserPassword` is a way of doing `AuthenticateUser` and that `SelectDateTime` is one of the steps of `DefineAppointment`.

- *isA*(`AuthenticateUserPassword`, `AuthenticateUser`)
- *isPartOf* (`SelectDateTime`, `DefineAppointment`)

Clearly, in order to draft a business process, having as a starting point the participating operations, these OR and AND relations do not suffice. The first step for this purpose is to define a sequence of operations offering the required functionality as a complex operation. Although this resembles the previously described AND hierarchy, in fact it is a combination of OR and AND relations introducing sequence constraints among the participating operations. In other words, a structure of this kind is characterised by a set of participating operations and edges connecting adjacent operations and modelling the control and data interactions between them. Such sequences of operations with well specified control and data flows between them essentially comprise workflows and therefore their investigation is subject of project Task 4.1 “Compliance Metamodel”. It is under consideration to port workflows in the Compliance Ontology in terms of worklets [5]; however, one has to balance the clarity and autonomy of models and foster separation of concerns.

Operations may take as input data to process and input parameters. The outcome of the operations may be some output data or alerts, or they may result to the execution of some other operation(s). Regarding the input and output data of an operation, two relevant relations can be specified corresponding to the data types that this operation can consume and produce respectively (cf. also Figure 3in Section 3.3):

- *hasInput*(*op*, $\langle dt \rangle^k$), where $op \in Op$, $\langle dt \rangle^k \subseteq \mathcal{P}(DT)$.
- *hasOutput*(*op*, $\langle dt \rangle^k$), where $op \in Op$, $\langle dt \rangle^k \subseteq \mathcal{P}(DT)$.

The required parameters for the execution of an operation are treated as attributes. For example, assuming an `encrypt` operation, the encryption algorithm and key size may be defined as attributes to the operation[3].

Moreover, operations are grouped following the mechanism reflected by the concept of *operations’ containers*. As aforementioned, this refers to components or other functional structures that typically offer a set of operations together. In that respect, the predicate that glues operations with a container type is the following:

[3] It should be noted though, that there may be multiple ways to model the same concept. For instance, in this example, an alternative way could be to particularise the `encrypt` operation to indicate the AES algorithm, i.e., `encryptAES` with *isA*(`encryptAES`, `encrypt`). Multiple modelling options reflect the flexibility of the BPR4GDPR approach, leaving the modeller to select the most convenient way in an ad hoc manner.

- *providesOperations*(*opct*, $\langle op \rangle^k$), where *opct* ∈ *OpCT*, $\langle op \rangle^k \subseteq \mathcal{P}(Op)$.

So far, operations have been thought of as abstract entities. Nevertheless, operations have their representation at the concrete level, notably by means of *Operation Instances*. The latter correspond to *implementations* or *instantiations* of operations by specific functional components. In Web Services' terms, an operation instance is equivalent to the abstract description of an operation, along with its concrete binding information, as derived by the associated operation container and the machine that provides the operation. Assuming, for instance, the operation `AppendFirewallRule` that reflects the functionality of appending one or more rules to a firewall's ruleset chain, the provision of this operation by two different firewalls, `Firewall#1`, `Firewall#2` ∈ *M*, implies two different elements in the *Operation Instances* (*OpI*) set.

For dealing with operation instances, several relations are defined, including the ones for associating an operation instance with the operation it instantiates and assigning operation instances to an operation container:

- *instantiatesOperation*(*opi*, *op*), where *opi* ∈ *OpI*, *op* ∈ *Op*.
- *containsOperationInstances*(*opc*, $\langle opi \rangle^k$), where *opc* ∈ *OpC*, ⟨opi⟩k ⊆ $\mathcal{P}$(*OpI*).

Nevertheless, not all operations can have corresponding instances; for instance, operations such as `execute`, `read` and `invoke` are considered only at the abstract level.

It is noted that as operations' containers are hosted in machines, the corresponding relation is defined, namely *hostsContainers*(*mt*, $\langle opct \rangle^k$), where *mt* ∈ *MT*, $\langle opct \rangle^k \subseteq \mathcal{P}(OpCT)$, allowing for the reflection of any effects of an operation performed on a machine to the hosted operations' containers and possibly to the operations they offer.

### 3.2.4 Events and context

In practice, access and usage control rules remain inactive until a set of conditions are fulfilled, i.e., until the conditions are evaluated and mapped to a truth value. The same typically applies as regards the execution of business processes. On the one hand, process execution is often triggered by an event; on the other hand, depending on contextual parameters, a process may vary as regards the actual execution pattern, e.g., in terms of contextual branching.

Therefore, we denote as *contextual* the authorisation policies containing dynamic authorisation provisions. In this regard, authorisation rules may depend on temporal contexts (e.g., authorisations granted only during working hours), geographical contexts (e.g., permission inside the physical boundaries of a company), *a priori* contexts (in which a permission to execute a set of actions can only be carried out as a result of the completion of previous actions). Thus, it is essential that not only the contextual conditions are captured by the model, but also that they are taken into consideration during the verification and transformation procedure, providing for the specification of context-based differentiated process behaviours, already during process formation.

Therefore, the model considers a *Context* (*Con*) definition set; a definition refers, on the one hand, to real-time parameters, such as location or time, while on the other, to events, such as system alerts. Context definitions leverage the evaluable conditions required to get a security rule to become applicable. Some examples of contexts are: *Temporal* contexts, which depend on the time at which the subject is requesting access to the system; *Spatial* contexts, which depend on the subject/object location; *Threat* contexts, which are activated on

the reception of a specific alert. Furthermore, structures of contexts are defined using two different relations, creating AND- and OR-trees, respectively. This way, a context *cont* ∈ *Con* can be expressed as the logical OR / logical AND of a set of sub-contexts, respectively, while negative contexts are also possible. Contexts that constitute combinations of sub-contexts are called *composite* contexts, as opposed to the *atomic* contexts.

As implied above, among the types of contextual parameters, a very important category concerns the ones that reflect events that have happened. In this context, the BPR4GDPR model also considers the respective set of *Event Types* (*ET*). Similar to contexts, it is characterised by the two relations, *isA* and *isPartOf*, creating AND- and OR-trees, respectively, whereas the two sets are related by means of the following relation, that maps a contextual parameter to the event type reflected by the former:

- *reflectsEvent*(*con*, *et*), where *con* ∈ *Con*, *et* ∈ *ET.*

A very fundamental concept for privacy that must anywise be taken into consideration is that of the *purpose* for which data are being collected, processed, and access to resources is requested. Due to its significance regarding privacy-awareness (cf. Section 2.2), it has been chosen not to be modelled as a special contextual parameter (as, for instance, in [6]), but as a stand-alone concept; this way, purpose is treated as a fundamental aspect that poses constraints applying *in principle*, and is distinguished from real-time parameters and events. Thus, the *Purposes* (*Pu*) set is defined, with its members forming hierarchies by means of OR (*isA*) relations modelling particularisation of a high level purpose to more specific ones; for instance, `MedicationPrescription` specialises the more general purpose `MedicalTreatment`.

Intuitively, not all operations can be used for fulfilling some purpose, in the sense that they are not compliant with and consistent to this purpose; this is expressed by means of the following relation:

- *compliantWithPurposes*(*op*, $\langle pu \rangle^k$), where *op* ∈ *Op*, $\langle pu \rangle^k \subseteq \mathcal{P}(Pu)$.

Through this relation and in combination with the aforementioned OR relation, it is implied that, apart from the explicitly defined purposes, an operation may serve more specific or more general ones. Additionally, all operations related to this by means of particularisation (OR) or inclusion (AND) relations are implied to also serve these purposes; however, as the distance between a child operation and the one which is explicitly bound to a purpose grows, the degree of affinity between the purpose and the child operation becomes weaker. This is particularly visible in cases of trivial operations that may serve almost every purpose. Finally, in order for a series of operations to be included in a complex structure thereof, such as a “worklet” [5], they must be proven to serve adequately at least one common purpose.

Likewise, the observation that not all roles may act for all purposes leads to the definition of the relation:

- *mayActForPurposes*(*r*, $\langle pu \rangle^k$), where *r* ∈ *R*, $\langle pu \rangle^k \subseteq \mathcal{P}(Pu)$.

It is noted that these two relations are explicitly specified —while they could be inferred by reasoning over rules— in order to support the purpose verification step of the process reengineering procedure (work of project Task 4.2 “Process verification and transformation”), allowing for a “quick” check of purpose compliance. In that respect, the *compliantWithPurposes* relation allows for checking whether the operations contained in a process are in line with the purpose that the process is supposed to serve. On the other hand, the *mayActForPurposes* relation serves for answering whether the roles held by the initiator of a process justify triggering the execution

of the process, in order for some specific purpose to be served; for instance, a `Doctor` should be able to execute a process serving the purpose of `MedicalTreatment`, while an `Accountant` should normally not.

Attributes constitute an essential part of the information model by complementing and further describing the members of the other sets at both abstract and concrete levels. Therefore, the *Attributes* (*Att*) set is defined and its members are formally described through the ordered pair ⟨*AttributeName*, *AttributeType*⟩. Regarding *AttributeType*, this can be some regular type, e.g., boolean, integer, etc., or a member of another set. For example, consider the following attributes:

- ⟨`Plain`, `boolean`⟩, denoting whether a set of data resource is at plain format, or some encryption mechanism has been applied. Here the *AttributeType* is set to `boolean`, implying that the attribute takes a boolean value.
- ⟨`WorksFor`, `Subcontractor`⟩, where `Subcontractor` ∈ *OrgT*, characterising some person, e.g., an external collaborator being granted with access to some corporate resources, by means of her relation to a company (she works for a subcontractor thereof). In this case, the `Subcontractor` element of the *OrgT* set is used as the *AttributeType*.

The mappings between entities and attributes are achieved through the following relations:

- *hasAttribute*(*entity*, ⟨*at*⟩$^k$), where *entity* ∈ *DT* ∪ *Op* ∪ *OpCT* ∪ *R* ∪ *MT* ∪ *OrgT* ∪ *D* ∪ *OpC* ∪ *U* ∪ *M* ∪ *Org* and ⟨*at*⟩$^k$ ⊆ $\mathcal{P}$(*Att*)., or
- *hasAttributeValue*(*entity*, *at*, *value*), including also the value of the associated attribute, where the value must be consistent with the attribute's type.

For example:

- *hasAttribute*(`Password`, `Plain`), where `Password` ∈ *DT*, `Plain` ∈ *Att*.
- *hasAttributeValue*(Server, `NetworkAddress`, 192.168.1.1), where Server ∈ *M*, `NetworkAddress` ∈ *Att* and 192.168.1.1 ∈ *D*.

It is noted that the attributes' value can be defined at both abstract and concrete levels. For instance, the value of the `Plain` attribute can be specified for the data type `AuthenticationCredentials` and consequently apply for the data types inheriting from this one. There exist also cases where the value has to be specified explicitly at the concrete level; an attribute `NetworkAddress` is defined for the machine type `NetworkInterface` but the value is unique for each machine of this type. We refer to attributes with their values defined already at the abstract level as *immutable* attributes; if this is not the case, the attributes are called *mutable* and their value can only be specified at the concrete level.

Moreover, attributes may be inherited from some entity to others through the various types of relations. However, there may exist attributes that should not be inherited, a case anticipated by defining a boolean propagation-related attribute on the attribute itself.

## 3.3 Information Model Ontology

Figure 3 illustrates the Information Model Ontology, providing the ontological —thus, also machine-readable, as well as semantic— implementation of the Information Model. All abstract concepts described in the previous

Sections comprise classes, whereas their intra- and inter-class relations are implemented as OWL object properties. It is to be noted that Figure 3 comprises an expanded view of Figure 2, since various entities are presented in addition, mostly of auxiliary nature.

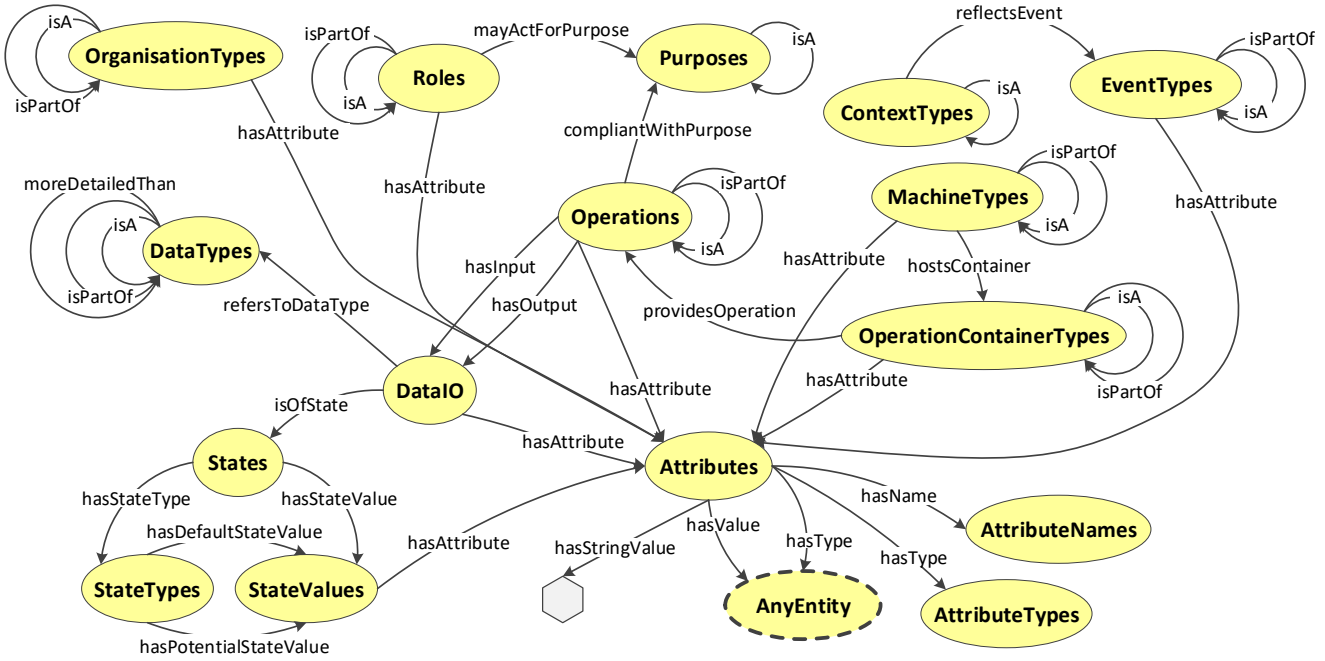


**Figure 3: Information Model Ontology**

To this end, the main classes comprising the BPR4GDPR Information Model Ontology are the following:

- `DataTypes`, which comprises all the types of data that can be used during the system's operation.
- `Roles`, which includes the roles that the system's users may hold.
- `Operations`, which contains all the operations that may take place.
- `OrganisationTypes`, reflecting the different types of organisations, external or internal, actual or virtual, that may be involved in operations.
- `OperationContainerTypes`, representing functional components that offer a set of operations together.
- `MachineTypes`, containing the types of machines that are used during the system's operation.
- `EventTypes`, reflecting the types of events that may affect the operations and/or may require some action to take place in response.
- `ContextTypes`, serving for the specification of run-time constraints.
- `Attributes`, providing for the description of properties that characterise the entities.

On the other hand, the main intra-class properties are `isA`, `isPartOf` and `moreDetailedThan` that, along with their inverses[4], essentially comprise AND- and OR- hierarchies, enabling inheritance, as well as dependencies specification. Associations between concepts of different classes are implemented by mean of inter-class OWL object properties; for instance, the roles that may act for a purpose are indicated by the

[4] Inverse properties are explicitly defined for all object properties in the ontology, in order to ease navigation from one ontological element to another.

`mayActForPurpose` property, whereas the attributes characterising a concept are related to the concept by means of the `hasAttribute` property.

In Figure 4, a simple and intuitively self-explained example of a `DataTypes` graph is illustrated, highlighting all three types of intra-class relations. On the other hand, Figure 5 provides an example of inter-class relations. It incorporates four classes, notably `Roles`, `Operations`, `Purposes` and `Attributes`, along with a few instances. In the example, `Speciality` is an attribute of the `Doctor` role[5]; the latter is in principle enabled to act for the `MedicalTreatment` purpose (through the `mayActForPurpose` property), that is also linked to the `ManagePrescription` operation, as compatible with said purpose. As regards this compatibility, it is noted that although the `compliantWithPurpose` property denotes the `ManagePrescription` operation as compatible, this is inherited also by the `CreatePrescription` operation, as a child of type `isA`, along with other operation types that are member of the sub-tree defined by `ManagePrescription` as root using the `isA` relation.

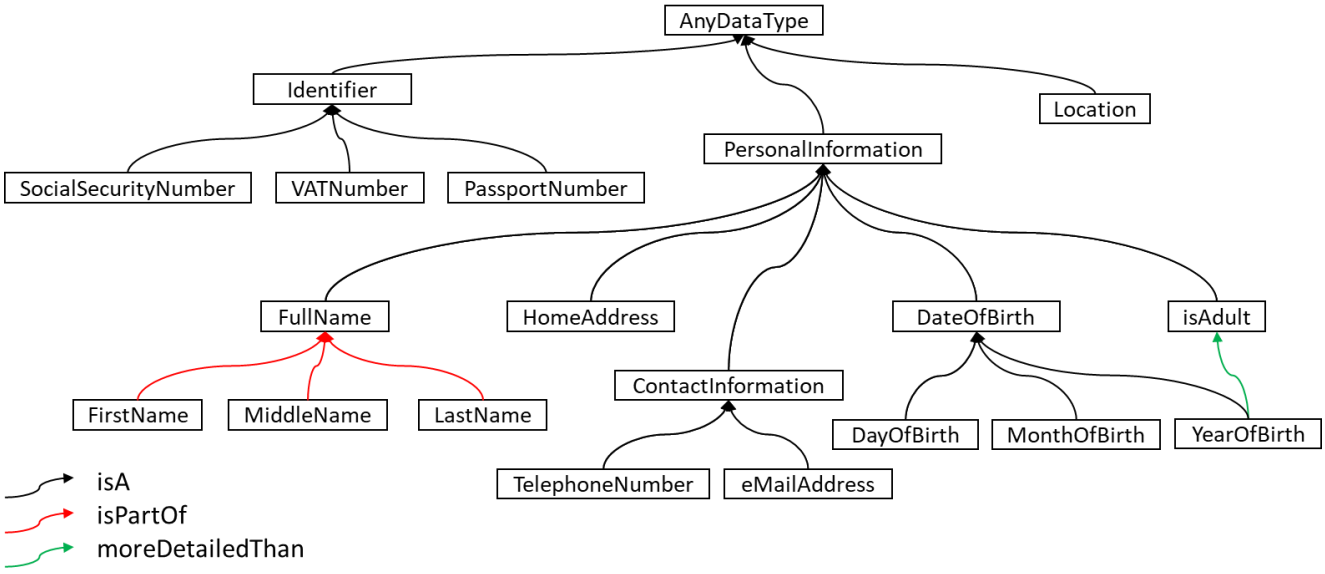


**Figure 4: Hierarchy example (`DataTypes` class)**

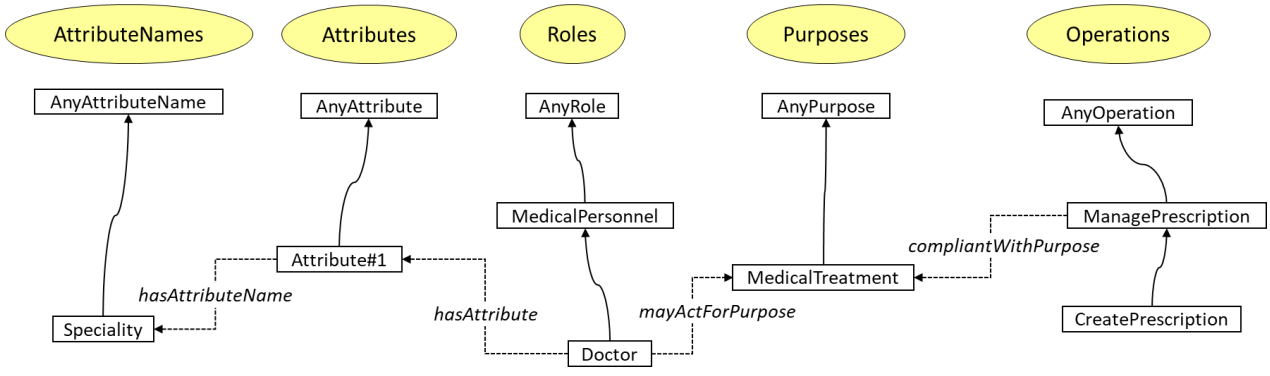


**Figure 5: Inter-class relations example**

[5] Details about how attributes are ontologically defined, including their name, type and value, are provided in Section 3.3.9.

### 3.3.1 DataTypes class

From an implementation point of view, members of the *DT* set constitute instances of the OWL `DataTypes` class and are organised into four hierarchies through the corresponding OWL object properties.

The generalisation-particularisation relationship, expressed in the model through the *isA*($dt_i$, $dt_j$) relation, is represented by the `isA` object property in the case of generalising a data type, and by its inverse in the opposite case. Similarly, the relationship regarding the detail level of the data forms another hierarchy and is implemented by means of the object property `moreDetailedThan` and its inverse one, namely `lessDetailedThan`. Moreover, the inclusion of a data type to another is expressed through the object property `isPartOf` and its inverse contains.

Finally, the basic hierarchy to which all the members of the class belong refers to and defines the inheritance relation among them and is expressed by means of the object property `inheritsFrom` and its inverse. Essentially, this is an implicitly derived property, based on the aforementioned three properties. The root of this hierarchy constitutes the `AnyDataType` instance (see also Figure 4) and every rule and attribute defined for it is consequently inherited to its descendants.

### 3.3.2 Roles class

As users of the system are assigned with roles, reflecting, e.g., their responsibilities inside an organisation, the corresponding set *R* was introduced. In that respect, the class `Roles` is specified in the ontology and the various roles (e.g., `Doctor` — cf. Figure 5) constitute instances of this class.

The members of the `Roles` class form two hierarchies, reflecting membership to a complex role and particularisation, respectively. The model's relation *isA*($r_i$, $r_j$) is translated to the object property `isA` which along with its inverse object property reflect the generalisation-particularisation relationship. Accordingly, the *isPartOf*($r_i$, $r_j$) relation is implemented by means of the object properties `isPartOf` and `contains`, which are inverse to each other. All role instances participate in a third hierarchy —having as root the individual `anyRole`— implied by the previous ones and reflecting the inheritance of characteristics, such as attributes and access control rules. For this purpose, the `inheritsFrom` object property is leveraged.

### 3.3.3 Operations class

The `Operations` class is defined within the ontology with its individuals corresponding to the various system operations, e.g., `Read`, `Anonymise`, `RequestConsent`, `CreatePrescription`, etc. Two hierarchies are formed with respect to the model's relations indicating different alternative operations that implement the same operation (*isA*($op_i$, $op_j$)) and granularity (*isPartOf*($op_i$, $op_j$)). In this context, the `isA` and `isPartOf` object properties, along with the corresponding inverse ones, implement these relations, whereas `inheritsFrom` fosters implicit inheritance of characteristics The most elementary instance from which all operations inherit rules and attributes is `anyOperation` (see also the example of Figure 5).

Regarding the input and output data of an operation, following the *hasInput*(*op*, $\langle dt \rangle^k$) and *hasOutput*(*op*, $\langle dt \rangle^k$) model's relations, we specify two relevant object properties, that is `hasInput` and `hasOutput`, respectively. The corresponding inverse object properties are also defined. The aforementioned object properties map operations to instances of an auxiliary class, namely `DataIO`, representing the data types that the given

operation consumes and produces, along with the information about their state (e.g., whether they are encrypted or not). In that respect, the following object properties are defined:

- `refersToDataType`, a property mapping a `DataIO` instance to the associated `DataTypes`.
- `isOfState`, a property mapping a `DataIO` instance to an instance of the class `States`.

Regarding the state representation in the ontology, three classes are defined, namely `States`, `StateTypes` and `StateValues`. `StateTypes` represent possible state types, such as `AnonymisationState` and `EncryptionState`. Instances of this class are associated with instances of the `StateValues` class, through the object properties `hasDefaultStateValue` and `hasPotentialStateValue`, denoting the default value of a state type and potential values of this type, respectively. For instance, the default value for the `AnonymisationState` is `NotAnonymised`. Additionally, state values may be associated with the attributes they bear by means of the object property `hasAttribute`. `States` instances serve for the definition of the state of a `DataIO` instance and are associated to a state type, a state value and some attributes (with the corresponding values). In that respect, the object properties `hasStateType`, `hasStateValue` and `hasAttribute` are defined.

Figure 6 illustrates how an operation is associated with its input and output, represented as `DataIO` instances. In the example, it is indicated that the `Anonymise` operation transforms any type of data (`AnyDataType`) from the `Plain` to `Anonymised` state.

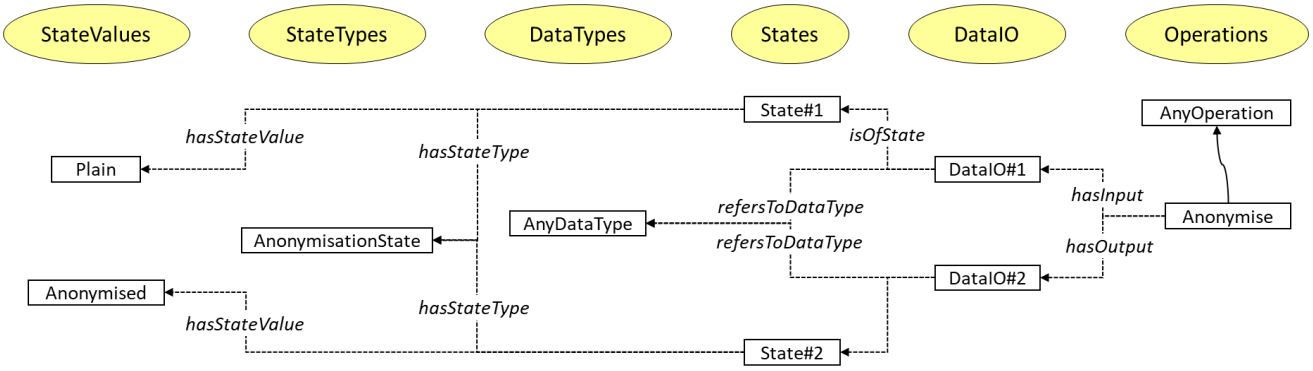


**Figure 6: Association of an operation to its input and output**

### 3.3.4 `OperationContainerTypes` class

The set *OperationContainerTypes* (*OpCT*) ontological implementation is realised by means of the class `OperationContainerTypes`. The object property `isA` denotes the generalisation-particularisation relation, whereas `isPartOf` reflects the inclusion relation; both imply the inheritance relation.

For gluing together operations with a container type, the `providesOperation` object property is leveraged, with operation containers types constituting its domain and operations its range. This is anticipated to significantly facilitate service discovery, particularly in the context of process re-engineering, when available tools and services are looked up towards incorporating data protection mechanisms.

Figure 7 illustrates an example of the `SQLDatabase` operation container type that provides the basic "CRUD" functions of persistence storage, i.e., `Create`, `Read`, `Update` and `Delete`. It is noted that the operations are

not directly related to the container, but inheriting the relation from their parent —in terms of the `isA` relation— operation `StorageFunction`.

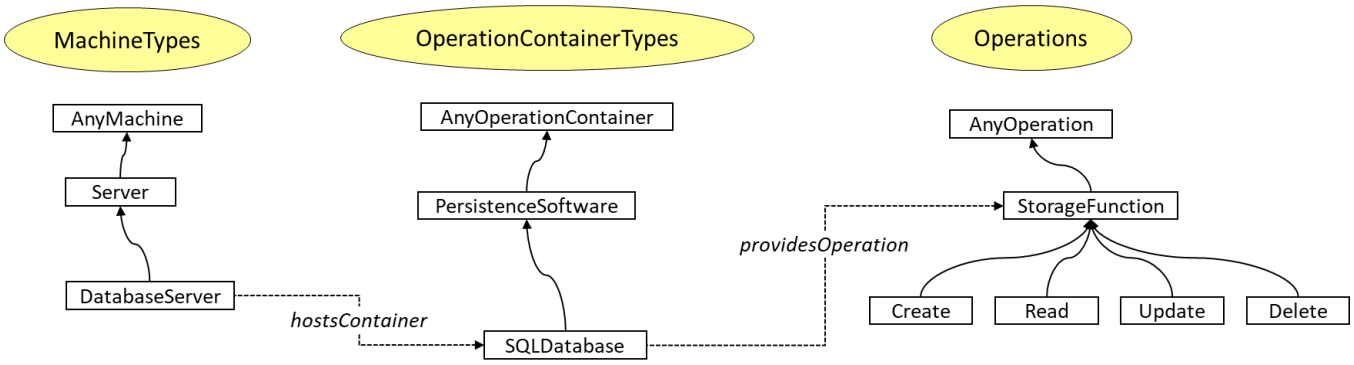


**Figure 7: Association of operations to containers and machines**

### 3.3.5 `MachineTypes` class

The `MachineTypes` class comprises the members of the abstract set *MachineTypes* (*MT*), e.g., `Server`, `Firewall`, `PersonalComputer`, etc. The object property `isA` reflects the generalisation - particularisation relation —as well as the inheritance of attributes and rules— among the various class individuals; on the other hand, `isPartOf` reflects inclusion. Machine types are also related to the `OperationContainerTypes` class members, i.e., the software deployed upon them, through the object property `hostsContainer`, in correspondence with the model's relation *hostsContainers*(*mt*, $\langle opct \rangle^k$), where $mt \in MT$, $\langle opct \rangle^k \subseteq \mathcal{P}(OpCT)$. For instance, a machine type `PersonalComputer` is related through this object property to an operation container type `OperatingSystem`. Similarly, Figure 7 highlights the hosting of an `SQLDatabase` by a `DatabaseServer`.

### 3.3.6 `OrganisationTypes` class

This class corresponds to the model's set *OrganisationTypes* (*OrgT*) and its individuals are organised by means of the object properties `isA` and `isPartOf`. It is noted that the overall pattern in the definition of organisation types may vary, and may result to types that are characterised by great heterogeneity. Indeed, this class is aimed at modelling all types of organisations, being actual (e.g., a company type) or virtual (e.g., the nodes of an ad hoc sensor network), external or internal (such as a department within the same company), an Authority or other public sector body, etc.

### 3.3.7 `EventTypes` and `ContextTypes` class

An important aspect of the BPR4GDPR approach is its context-awareness; context is a fundamental parameter of rules, whereas context is ultimately considered as an important parameter during the procedure of process verification and transformation, often resulting in the introduction of contextual branching patterns within the BPR4GDPR workflows and, consequently, their respective handling at execution time.

There is a variety of contexts that can play a role in an environment like BPR4GDPR. Therefore, an important requirement for the implementation of context-awareness within BPR4GDPR has been flexibility, in the sense of enabling to support, to the extent possible, nearly any type of context. In that respect, the adopted approach puts in place a quite generic and flexible framework, fostering the potential enhancement of the model with additional contextual types, apart from the ones comprising the default BPR4GDPR configuration or required by the project use cases and trials.

The central ontological class for context-awareness in BPR4GDPR is `ContextTypes`. The instances of the `ContextTypes` class serve for the specification of contextual expressions and model cases like the following:

- The time is within working hours
- The data server is located in Europe
- A breach has been detected.
- The DPO has provided her approval.
- The process has been initiated by a senior manager.

As also reflected by the examples above, context types, e.g., `TimeIs` or `BreachDetected`, constitute the subject of contextual expressions and are associated with a value, such as "15:12", "Paris" or "YES". Moreover, it should be stressed that whether an event has been taken place or not (as the case of a data breach), as well as various metadata related to the event (e.g., when the event has taken place) constitute also contextual types. Therefore, the `ContextTypes` class is complemented by the `EventTypes` class, containing the semantic types of events. To this end, the `reflectsEvent` ontological property is leveraged.

For organising context types, we define sub-classes of the class `ContextTypes`, such as `SpatialContextTypes`, `LocationContextTypes`, `HistoricalContextTypes`, etc.

### 3.3.8 `Purposes` class

The various purposes for which data are collected and processed and, to this end, access to the data is requested constitute the members of the `Purposes` class and are organised by means of the object property `isA`, through which inheritance of rules can be inferred.

Purposes are additionally related with individuals of the `Roles` and `Operations` classes; following the model's relations *mayActForPurposes*$(r, \langle pu \rangle^k)$ and *compliantWithPurposes*$(op, \langle pu \rangle^k)$, the corresponding object properties are defined, namely `mayActForPurpose` and `compliantWithPurpose`, mapping purposes to roles and operations, respectively.

### 3.3.9 `Attributes` class

As aforementioned, attributes are characterised by a name and a type, while they are either mutable or immutable, depending on whether they have a value or not. An attribute can be assigned to an abstract entity without its value being specified, just to denote that the said entity will bear this attribute until the latter gets a value at the concrete level. For instance, all operations bear an attribute named `Output`, but its value (and not the type of the value) is "visible" only for each operation instance; or, an attribute named `NetworkAddress` can be defined for the machine type `NetworkInterface` but the value is unique for each machine of this type. On the other hand, whether a role is executive or not is specified already at the abstract level and thus, the attribute with the name `Executive` is associated to the corresponding value.

In that respect, we define the following classes:

- `AttributeNames`, reflecting the attributes that can be defined and containing instances like, e.g, `Executive`, `NetworkInterface`, etc.
- `AttributeTypes`, which encapsulates the types that are not contained in another class of the Information Model, e.g., `String`, `Integer`, or a concept defined in an external ontology, such as `Colour`.

- `Attributes`, containing all the instances that complement other concepts of the ontology; these instances are always associated with a name and a type and, in some cases, with a value.

For the association of an entity with an attribute, the `hasAttribute` object property is leveraged, which points at an `Attributes` instance, whereas for the association of the instances belonging to the aforementioned classes, the following properties are leveraged:

- `hasName`, a functional object property associating an `Attributes` instance with its name, i.e., an `AttributeNames` instance.
- `hasType`, a functional object property associating an `Attributes` instance with its type, i.e., an `AttributeTypes` instance or an instance of another class of the Information Model Ontology.
- `hasValue`, a functional object property mapping an `Attributes` instance to another ontological instance constituting its value.
- `hasStringValue`, a functional datatype property mapping an `Attributes` instance to a String value.

Intuitively, the latter two properties (`hasValue` and `hasStringValue`) are meaningful only for the case of immutable attributes, since mutable attributes take no value at this level.

In Figure 8, two instances of the class `Attributes` are defined, named through the property `hasName` that points at the `AttributeNames` instances `isSensitive` and `Speciality`, respectively. `Attribute#1` is bound with the type `String` (instance of the class `AttributeTypes`) by means of the `hasType` property, whereas, being an immutable attribute, bears also a value (`"YES"`).

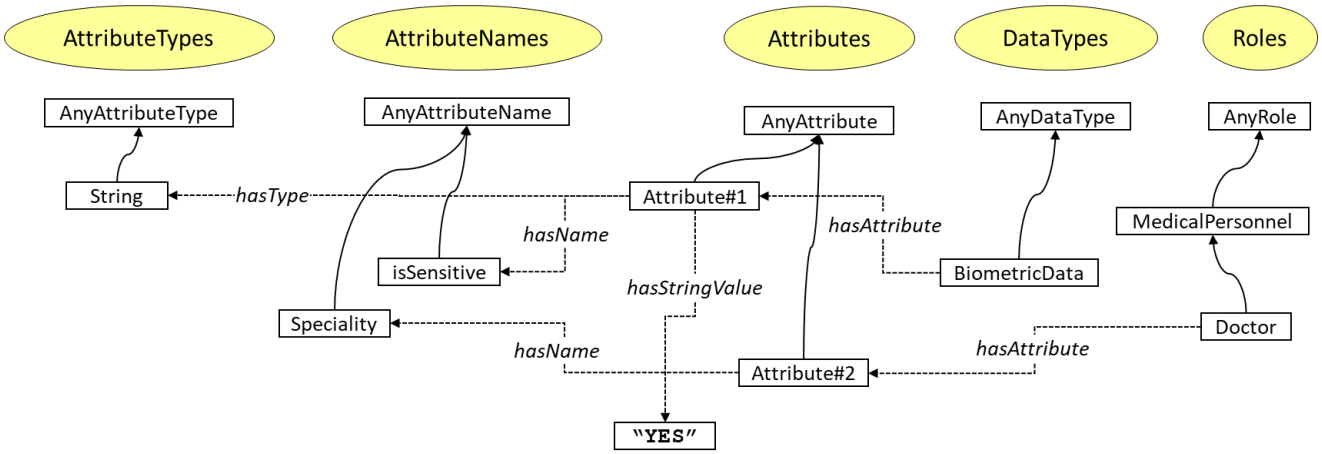


**Figure 8: Mutable and immutable attributes**

## 3.4 Default Instances

The content of the Information Model Ontology largely depends on the specific needs of each organisation adopting the BPR4GDPR solutions; intuitively, as regards, e.g., data types and operations, each sector has its own typical data sets and functions —consider, e.g., healthcare vs. airlines—, whereas each organisation has its own internal structure in terms of departments, roles, machinery and software systems. To this end, it is anticipated that each organisation will create its own configuration.

However, the project aims at providing a "default configuration". This consists in typical instances that are more or less present in all organisations that process personal data (e.g., `FullName`). Some indicative instances for the basic classes of information are provided below, in Table 2.

Thereupon, each organisation should create its own model. Indeed, also for the BPR4GDPR use cases, this work is already being done in the context of project Task 6.3 "Trials and validation", particularly the activity referred to as "Data protection validation".

Furthermore, an idea that emerged in the context of the project is to enable the enhancement of the Information Model Ontology by means of *stencils*, i.e., sets of information instances that shall be imported. Following this approach, a hospital, for example, could download and import in the model the "healthcare stencil", consisting of a rich set of elements that are specific to the healthcare domain. This could also contain the appropriate rules' set that would typically apply in such environments. This would further facilitate organisations in adapting to the GDPR needs and requirements.

| Data Types | |
|---|---|
| Identifier | CustomerNumber |
| SocialSecurityNumber | IdentityNumber |
| PassportNumber | VATNumber |
| FullName | FirstName |
| LastName | Middlename |
| DateOfBirth | YearOfBirth |
| MonthOfBirth | DayOfBirth |
| Age | isAdult |
| Gender | Race |
| Nationality | Citizenship |
| MaritalStatus | Nickname |
| HomeAddress | HomeAddressCountry |
| HomeAddressStreet | HomeAddressNumber |
| HomeAddressPostCode | HomeAddressApartmentNumber |
| ContactInformation | TelephoneNumber |
| eMail | Location |
| Profession | Title |
| WorkAddress | WorkAddressCountry |
| WorkAddressStreet | WorkAddressNumber |
| WorkAddressPostCode | WorkAddressApartmentNumber |
| Employer | CookieID |
| CreditCardNumber | CreditCardExpirationDate |
| CreditCardSecurityID | Salary |

| | |
|---|---|
| AnnualIncome | ContractNumber |
| Photo | Video |
| Username | Password |
| **Operations** | |
| Read | Write |
| Execute | Update |
| Delete | Store |
| Log | Contact |
| Collect | Access |
| Record | Organise |
| Structure | Adapt |
| Alter | Retrieve |
| Consult | Use |
| Disclose | Disseminate |
| Align | Combine |
| Restrict | Erase |
| Destruct | Profile |
| Copy | Mirror |
| Export | Import |
| Port | Assess |
| Anonymise | Pseudonymise |
| Sign | DigitallySign |
| Encrypt | Decrypt |
| Authenticate | Authorise |
| **Roles** | |
| DataSubject | DataProtectionOfficer |
| Customer | User |
| Employee | Manager |
| ChiefExecutiveOfficer | ManagingDirector |
| ChiefFinancialOfficer | ChiefOperationsOfficer |
| ChiefSecurityOfficer | SecurityOfficer |
| ChiefComplianceOfficer | ComplianceOfficer |
| BoardOfDirectors | LegalRepresentative |
| Administrator | SystemAdministrator |
| ITManager | ITAdministrator |

| | |
|---|---|
| Consultant | LegalConsultant |
| TaxConsultant | Agent |
| Freelancer | Counterparty |
| ContractWorker | Accountant |
| **Organisation Types** | |
| DataProcessor | DataController |
| DataProtectionAuthority | PublicBody |
| TaxAuthority | Ministry |
| Partner | ThirdParty |
| Subsidiary | HoldingCompany |
| ParentCompany | Department |
| InternalUnit | BusinessUnit |
| LegalDepartment | FinancialDepartment |
| MarketingDepartment | HumanResourcesDepartment |
| CustomerCareDepartment | CustomerServicesDepartment |
| RetailDepartment | eShopDepartment |
| ITDepartment | WorkAgency |
| InsuranceCompany | Bank |
| ServiceProvider | PaymentServiceProvider |
| UtilityOperator | TelecomOperator |
| Vendor | Client |
| MarketingAgency | Advertiser |
| **Operation Container Types** | |
| ITSystem | SoftwareSystem |
| OperatingSystem | CloudPlatform |
| PersistenceSoftware | Database |
| SQLDatabase | NoSQLDatabase |
| FileManagementSystem | DocumentManagementSystem |
| CRMSystem | ERPSystem |
| DigitalMap | IntegratedSystem |
| WorkflowManagementSystem | EnterpriseServiceBus |
| WebPortal | ApplicationServer |
| UserManagementSystem | HRMSystem |
| Firewall | IntrusionDetectionSystem |

| Machine Types | |
|---|---|
| Server | PersonalComputer |
| Laptop | SmartPhone |
| Tablet | Phone |
| HardDrive | RemovableMedia |
| Printer | Camera |
| Scanner | BiometricScanner |
| FingerprintScanner | CardReader |
| BarcodeReader | RFIDReader |
| Router | Switch |
| GPSLocator | Vehicle |
| **Event Types** | |
| DataBreach | SecurityIncident |
| IntrusionIncident | SystemFailure |
| ProcessInitiated | UserAuthenticated |
| TaskExecuted | DataRetrieved |
| DataAccessed | DataSent |
| DataDeleted | DataCorrected |
| ConsentProvided | ConsentDenied |
| ConsentRevoked | NotificationSent |
| RetentionPeriodExpired | DataSubjectRequestReceived |
| DataSubjectRequestReplied | ProcessCompleted |
| ProcessAborted | TaskAborted |
| TimeExpired | |
| **Context Types** | |
| Temporal | Spatial |
| Historical | Time |
| Location | EventDetected |
| DataBreachDetected | SecurityEventDetected |
| VideoSurveillanceOn | |
| **State Types** | |
| Encrypted | Plain |
| Anonymised | Pseudonymised |
| Accessed | Updated |

| | |
|---|---|
| Deleted | |
| **Attributes** | |
| isSensitive | SensitivityLevel |
| Schedule | isExecutive |
| KeyLength | Algorithm |

**Table 2: Indicative default instances in the Information Model Ontology**

# 4 Policy specification framework

In general, policies are aimed at regulating actions, by putting in place appropriate behavioural norms that actions should follow. To this end, BPR4GDPR defines rules over *actions*; an action reflects anything that takes place during the function of a system, and can be seen as an *operation* of an *actor* over a *resource*, possibly within an *organisation*, the latter being actual or virtual, internal or external.

But, in order to effectively regulate the operation of a complex system, security and privacy policies, as well as access and usage control rules must incorporate additional features. First, there is the requirement of full abstraction potential, meaning that all concepts are described at a fully abstract level; this enables cohesively treating entities falling under the same conceptualisation. In that respect, Role-Based Access Control (RBAC) [7] first introduced the *role* abstraction for users, which is the typical pattern followed by most approaches, whereas the current —and still emerging— standard is the Attribute-Based Access Control (ABAC) [8], that is based on abstractions of entities' attributes. However, only very few approaches support representation using abstractions for all elements, while the use of hybrid approaches is very limited.

Second, policies must contain norms regarding what should have happened before and what should provisionally happen next, as well as be bound to events and context, as described also in Section 2.2. While much research has been done in the area of contextual security and privacy policies, most models only partially support some straightforward contexts, especially temporal, spatial, and related to history [9]. Nevertheless, complex systems should be able not only to behave taking into account contextual parameters, but also to base their operation on events. In other words, events taking place "somewhere" and "somewhen" are interpreted as context whereas the contextual changes should be promptly available to any affected entity and trigger contextual changes that will result in functional adaptation.

Taking the above into consideration, each {*actor*, *operation*, *resource*, *organisation*} quadruple is characterised as an *action*. Intuitively, actions may represent whatever takes place in the course of time and they can be used for modelling past, present and future context, as well as events. Therefore, rules pertaining to a policy need to be defined over constructions such as {*purpose*, *context*, *action*, *pre-action*, *post-action*}, surrounded by the appropriate predicates that describe permissions, prohibitions and obligations of the policy enforcing entities. Having a deeper look to this pattern highlighted in Figure 9:

- *action* describes the core of the rule, i.e., what action the rule by definition permits, prohibits or obliges to take place.
- *purpose* reflects the overall objective behind data collection and/or processing; in fact, an operational decision cannot be taken regardless of this parameter, which can significantly differentiate an entity's behaviour.
- *pre-action* reflects actions that should have previously taken place in order for the rule to be activated; as an example, there can be the case where a patient should have provided explicit consent (pre-action) in order for her medical record to be processed for research purposes.
- *post-action* similarly implies anything that needs to take place after the enforcement of a rule; for instance, a rule may permit reading some data for providing a service, but may additionally require that the data are deleted immediately after.
- *context* describes conditions defined over "environmental" properties and states, as well as events.

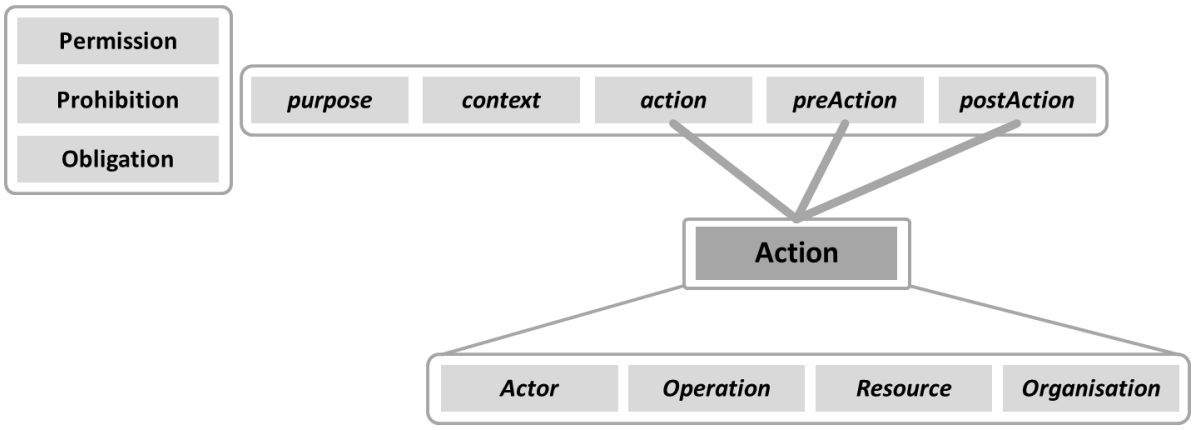


Figure 9: BPR4GDPR rules format

All these elements should be enabled to be combined using logical and domain-specific descriptive operators, in order for complex compositions to be specified. In process terms, this can be leveraged for the definition of complex structures of tasks, sometimes referred to in the workflows' terminology as "worklets" [5]. Enhanced with the appropriate semantics (e.g., temporal), even with some native fuzziness, the expected behaviour of a complex system of actions can be packaged as a worklet to be part of the knowledge base and be consequently checked and verified against the reported behaviour.

## 4.1 Access and usage control rules

As said above, an action refers to the situation where an *actor* performs an *operation* on a *resource*, possibly within an *organisation*. Whereas what can be defined as an operation or as an organisation with respect to the Information Model (Section 3) is rather straight-forward (since the ontology thereof includes correspondingly named classes), different types of entities may play the role of actors and resources, thus be members of the corresponding *Actors* (*A*) and *Resources* (*Res*) sets. Although there are some obvious patterns (e.g., users are typically actors and data are always resources), which entities can be actors and/or resources strongly depends on each individual organisation, and operational aspects and modelling choices thereof.

In BPR4GDPR, an action is defined as follows:

*Definition 1*. An action $act \in Act$ is a tuple $\langle a, op, res, org \rangle$, such that: $a \in A$ is an actor; $op \in Op$ is an operation; $res \in Res$ is a resource; and $org \in Org$ is the organisation within which an action takes place.

An action can be either *atomic* or *composite*, depending on whether the associated operation can be decomposed to more elementary operations or not, following the hierarchical relations in *Op*. Actions are also categorised to *abstract*, *concrete* and *semi-abstract*, depending on whether actors and resources are defined at abstract, concrete or mixed level.

Further, it should be stressed that the elements of an action can be specified as *enhanced entities* that include, apart from the entity's semantic type, expressions over its attributes and/or sub-concepts, thus refining the concept definition, towards specifying attribute-based constraints and access and usage control rules.

Upon the concept of actions, access and usage control rules are specified; they are defined as follows:

*Definition 2*. An *access and usage control rule* is a structure:

$$\left.\begin{matrix} Permission \\ Prohibition \\ Obligation \end{matrix}\right\}(pu, act, preAct, cont, postAct)$$

where $act \in Act$ is the action that the rule applies to; $pu \in Pu$ is the purpose for which $act$ is permitted/prohibited/obliged to be executed; $cont \in \mathcal{P}(ConT)$ is a structure of contextual parameters; $preAct \in Act$ is a structure of actions that should have preceded; $postAct \in Act$ refers to the action(s) that must be executed following the rule enforcement.

## 4.2 Default rules

In the following, the fundamental rules that should regulate a BPR4GDPR-compliant system are identified; they are directly derived from the GDPR. They are meant to constitute the default configuration of the BPR4GDPR ruleset, i.e., the minimal set of rules before the policy framework is adapted to the needs of an organisation that adopts the solution. To this end, they are characterised by the principle of defining the constraints at the most strict level —e.g., that *any data access and use should by default be forbidden* (cf. Rule #1)—, with necessary permissive rules needing to be defined as exceptions to this principle.

Rule #1. Nobody should have access to any data (secrecy by default)

Rule #2. An authorisation mechanism should be available and used in any access and use operation

Rule #3. A data breach must be reported within 72 hours starting when the controller has ascertained the breach

Rule #4. No data should be shared to third parties

Rule #5. If shared, data must be pseudonymised or anonymised

Rule #6. Stored data must be encrypted

Rule #7. Data subjects should have access to their data

Rule #8. No data processing should take place unless the data subject has provided consent

Rule #9. The means for updating consent must be provided, thus keeping consent information up to date

Rule #10. An easy way to revoke consent must be provided

Rule #11. If consent has been obtained, it should be tracked

Rule #12. Data subjects must be provided with adequate privacy notice

Rule #13. Data subject must be able to update or modify his/her own personal data

Rule #14. Personal data should be stored for no longer than required by the underlying law/regulation, as well as the duration of the purpose of the processing itself

Rule #15. The DPO should have broad access to personal data processing activities

Rule #16. The DPO should be involved in any data processing assessment

Rule #17. No actions may be taken without the prior consultation of the DPO

Rule #18. Access to personal information should be granular, according to the role and the functions of the person who is meant to access

Rule #19. Features to carry out data portability should be enabled

Rule #20. The system should allow to delete only personal data eventually considered as either not relevant for specific and imperative obligations of the controller or inaccurate

Rule #21. Any record of processing for each controller should be stored separately and accessed only by authorised users

Rule #22. The system should allow for the customisation of data retention periods according to the particularities of the collected data

Rule #23. Transfer of personal data must be tracked

Rule #24. System administrator's logging activity should be tracked

In the context of the project Task 3.2 "Rule-based access and usage control", work is being performed towards providing the formalisation of these rules.

# 5 Conclusions

This Deliverable documented the Compliance Ontology, aiming at providing a high-level codification of the GDPR, by extracting the concepts that need to be addressed by the BPR4GDPR policy framework, as well as by the privacy-aware process reengineering. In that respect, the work performed in the context of project Task 3.1 "Compliance Ontology", reflected by the Deliverable, has investigated:

- Various technical aspects stemming from the GDPR that constitute the challenges that the BPR4GDPR solutions, particularly the access and usage control rule-based framework, must face and address
- The concepts comprising the fundamental ontology of compliance
- The categories of these concepts, i.e., the ontological classes, as well as the way these concepts are organised
- The patterns characterising the relations between these concepts
- The very basic entities implementing the concepts, i.e., the default instances of the ontology
- The foundations for the instantiation of the Compliance Ontology by means of Semantic Web technologies, notably the Web Ontology Language (OWL)
- The generic rules governing the BPR4GDPR ecosystem as implied by the Compliance Ontology

The Compliance Ontology achieves its goals, in the sense that it fosters meeting the identified challenges. For example —anticipating the work of legal assessment within project Task 6.1—, a very fundamental concept for privacy —and a basic principle of the GDPR— is the *purpose* for which data are processed, being a core part of the lawfulness principle. In the ontology, purpose plays indeed an important role; the ontology explicitly includes the *Purposes* (*Pu*) set, whereas purpose is an integral part of the rules and a parameter that affects separation and binding of duty. In addition, the purpose principle prescribes mechanisms for specifying the compatibility between processing purposes, while also providing for checking whether the processing purposes are consistent with those for which data have been collected. Therefore, BPR4GDPR ontology provides the means for defining prevention rules regarding incompatible purposes; indeed, the formal conceptualisation of purposes, by means of the *Pu* set, has been considered, while for their direct association with data processing activities, the *compliantWithPurpose* relation implements the association with operations. Moreover, the BPR4GDPR model provides the means for the specification of the purposes that a role may hold (*mayActForPurposes* relation), as well as for checking compliance between a role's acting purpose and the ones served by an operation. All these are complemented by well-defined norms for being inherited across the corresponding graphs of the concepts involved.

The principles of necessity, adequacy and proportionality are also tightly connected to purpose. In fact, the project provides the means for necessity specification, as well as for examining whether the collection or processing of specific data is necessary for the provision of the service in question. In that respect, the adopted pattern for rules' specification implements a relation between data, processing activities, roles and purposes, thus enabling the definition of necessity and proportionality constraints. Due to the rich descriptiveness of the rules, these constraints are extended to past and future actions, contextual conditions and the organisation within which the actions are performed.

Furthermore, the proposed ontology enables the definition of different levels of data granularity, so that the accuracy of disclosed data can be adjusted depending on the purpose and the subject requesting access to the said data, among others. In that respect, the adopted approach puts in place the means for the definition of

different AND- and OR- hierarchies, describing inclusion and particularisation, as well as a relation explicitly denoting the detail level (*moreDetailedThan* relation). This way, the presented approach enables the description of concepts in a manner that is significantly more fine-grained than the other works in the field. This is complemented also by the approach taken for the description of *attributes*, providing for the association of concepts with their characteristics, as well as the very flexible pattern adopted for the encapsulation of functions into *containers*., without underestimating the role of machinery.

The described Compliance Ontology comprises the first result of the Work Package 3, providing the ground for the rule-based access and usage control framework of the project to be developed. In this context, the ontology will also be subject to evolution, based on the findings not only of the work within this Work Package, but also the assessment activities. Therefore, it is expected that a revised version of the ontology will be available at the end of the project Task 3.1 duration, i.e., month M25.

## References


[1] Solove, D. J. (2006). A Brief History of Information Privacy Law. In C. Wolf (Ed.), *Proskauer on Privacy: A Guide to Privacy and Data Security Law in the Information Age* (pp. 1-46). Practising Law Institute.

[2] Lioudakis, G. V., Gaudino, F., et al. (2010). Legislation-Aware Privacy Protection in Passive Network Monitoring. In I. M. Portela, M. M. Cruz-Cunha (Ed.), *Information Communication Technology Law, Protection and Access Rights: Global Approaches and Issues* (pp. 363–383). New York: IGI Global Pubs.

[3] Gutwirth, S., Leenes, R., De Hert, P., & Poullet, Y. (Eds.). (2013). *European Data Protection: Coming of Age*. Berlin, Germany: Springer.

[4] Papagiannakopoulou, E. I., Koukovini, M. N., et al. (2015). Privacy-Aware Access Control. In M. Khosrow-Pour (Ed.), *Encyclopedia of Information Science and Technology* (pp. 4403-4411). IGI Global.

[5] Adams, M., ter Hofstede, H. M. A., Edmond, D., & van der Aalst, W. M. P. (2006). Implementing Dynamic Flexibility in Workflows using Worklets. *Technical Report BPM-06-06.* BPM Center.

[6] Ajam N., Cuppens-Boulahia N., Cuppens F. (2010). Contextual Privacy Management in Extended Role Based Access Control Model. In: Garcia-Alfaro J., Navarro-Arribas G., Cuppens-Boulahia N., Roudier Y. (eds) Data Privacy Management and Autonomous Spontaneous Security. DPM 2009, SETOP 2009. Lecture Notes in Computer Science, vol 5939. Springer, Berlin, Heidelberg

[7] Ferraiolo, D.F., Sandhu, R., Gavrila, S., Kuhn, R.D., & Chandramouli, R. (2001). Proposed NIST Standard for Role-Based Access Control. *ACM Transactions on Information and System Security*, *4*(3), 224-274.

[8] Hu, V. C., Ferraiolo, D., Kuhn, R., et al. (2013). Guide to Attribute Based Access Control (ABAC) Definition and Considerations. *NIST special publication 800-162*, NIST.

[9] Cuppens, F., & Cuppens-Boulahia, N. (2008). Modeling contextual security policies. *International Journal of Information Security, 7*(4), 285-305. Berlin, Germany: Springer-Verlag.