Primer

The Data Privacy Vocabulary [[DPV]] enables expressing machine-readable metadata about the use and processing of personal data based on legislative requirements such as the General Data Protection Regulation [[GDPR]]. This document acts as a 'Primer' for the DPV by introducing its fundamental concepts and providing examples of use-cases and applications. It is intended to be a starting point for those wishing to use the DPV and an orientation for people from all disciplines. The canonical URL for DPV is https://w3id.org/dpv# which contains its specification.

This primer document aims to ease adoption of DPV by providing:

- A high-level conceptual explanation of the DPV and its modelling of concepts;
- Self-contained examples that illustrate how the concepts and data models provided by DPV can represent information associated with personal data handling; and
- Guidance towards application of DPV in use-cases and technologies.

DPV Family of Documents

- [[DPV-Primer]]: The Primer serves as an introductory document to DPV and provides an overview of its concepts.
- [[DPV]]: (This document) The DPV Specification is the formal and normative description of DPV and its concepts. It provides a serialisation of the concepts as a taxonomy using SKOS.
- Extensions to Concepts:
  - [[DPV-GDPR]]: Extension to the DPV providing concepts relevant to [[GDPR]].
  - [[DPV-PD]]: Extension to the DPV providing a taxonomy of personal data categories.
  - Serialisations of DPV:
    - [[DPV-SKOS]]: A serialisation of the DPV using [[RDFS]] and [[SKOS]] to enable its use as a schema or ontology.
  - [[DPV-OWL]]: is a serialisation of the DPV using [[OWL]] to enable its use as an ontology.
- [[DPV-NACE]]: [[NACE]] taxonomy serialised in RDFS

Related Links

- For a general overview of the Data Protection Vocabularies and Controls Community Group [[DPVCG]], its history, deliverables, and activities - refer to DPVCG Website.
- The peer-reviewed article "Creating A Vocabulary for Data Privacy" presents a historical overview of the DPVCG, and describes the methodology and structure of the DPV along with describing its creation. An open-access version can be accessed here, here, and here.

Format for examples

Contributing to DPV The DPVCG welcomes participation regarding the DPV, including suggestions for primer, expansion or refinement of DPV terms, addressing open issues, and other relevant matters.

While we welcome participation via any and all mediums - e.g., via GitHub pull requests or issues, emails, papers, or reports - the formal resolution of contributions takes place only through the DPVCG meeting calls and mailing lists. We therefore suggest joining the group to participate in these discussions for formal approval.

For contributions to the DPV, please see the section on GitHub. The current list of open issues and their discussions to date can be found at GitHub issues.

This document is a 'draft version' and is intended for feedback and comments.

Introduction

The [[[DPVCG]]] was formed in 2018 through the [[[SPECIAL]]] with the ambition of providing a machine-readable and interoperable vocabulary for representing information about the use and processing of personal data, whilst inviting perspectives and contributions from a diverse set of stakeholders across computer science, IT, law, sociology, philosophy – representing academia, industry, policy-makers, and activists. It identified the following issues through the W3C Workshop on Privacy and Linked Data:

a. lack of standardised vocabularies to represent information about use and processing of personal data;
b. lack of descriptive taxonomies that describe purposes of processing personal data which are not restricted to a particular domain or use-case; and
c. lack of machine-readable representations of concepts that can be used for technical interoperability of information.

The outcome of addressing these resulted in the creation of the [[[DPV]]], which provides a vocabulary and ontology for expressing information related to processing of personal data, entities involved and their roles, details of technologies utilised, relation to laws and legal justifications permitting its use, and other relevant concepts based on privacy and data protection. While it uses the EU's [[[GDPR]]] as a guiding source for the creation and interpretation of concepts, the ambition and scope of DPV is to provide a broad globally useful vocabulary that can be extended to jurisdiction or domain specific applications.

People, organisations, laws, and use-cases have different perspectives and interpretations of concepts and requirements which cannot be modelled into a single coherent universal vocabulary. The aim of DPV is to act as a core framework of ‘common concepts’ that can be extended to represent specific laws, domains, or applications. This lets any two entities agree that a term, for example, `PersonalData`, refers to the same semantic concept, even though they might interpret or model it differently within their own use-cases.

Using DPV

The motivation of DPV is to provide a '*data model*' or a '*taxonomy*' of concepts that act as a vocabulary for the interoperable representation and exchange of information about personal data and its processing. For this, the DPV specification represents an abstract model of concepts and relationships that can be implemented and applied using technologies appropriate to the use-case's requirements. This specification is serialised using [[SKOS]] to produce a formal documentation of its contents.

DPV Serialisations

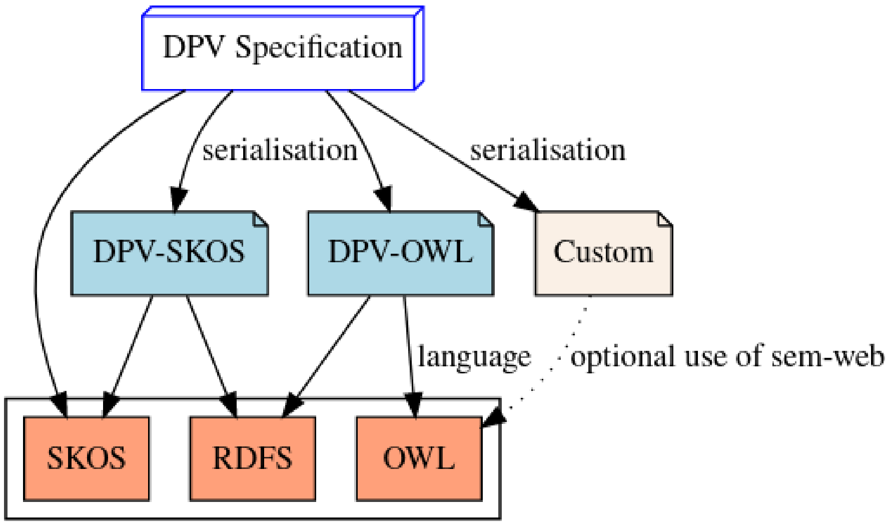

Figure 1: Serialisations of DPV specification

The DPV is serialised using [[RDF]] to provide a formal interoperable and machine-readable representation of information. While this enables its use as a semantic web vocabulary, the DPV can also be used without (or alongside) semantic web by either utilising a format such as [[JSON-LD]] that retains the semantics and provides convenience of using JSON, or through other formats such as a CSV or a flat-list of concepts. This section provides an overview of such approaches where DPV can be used both with and without semantic web.

The following are four (non-exhaustive) ways DPV can be used based on the requirements of an use-case. For guidance on how to adopt DPV concepts within an use-case, refer to [[[DPV GUIDES]]].

1. As a **taxonomy or collection of concepts**: The [[DPV]] specification provides a vocabulary of concepts (e.g. `Purpose`) and relationships (e.g. `hasPurpose`) without providing any restrictions on their usage (e.g. property range assertions). This specification can be used in cases where only the concepts within DPV are needed (e.g. as a list or hierarchy of purposes), either in RDF or as 'flat lists' or CSV files.
2. As a **'schema' or 'lightweight-ontology'**: The [[DPV-SKOS]] is a serialisation of the [[DPV]] specification that provides a lightweight ontology for modelling or annotating information. For this, it uses [[SKOS]] to represent the concepts and [[RDFS]] to model relationships between them. This serialisation can be used in cases where the DPV is to be used as a 'data model' or 'schema' without formal logical assertions. It is suitable in cases where simple(r) inferences are required, or where the strict interpretation or restrictions of OWL are not needed, or the rules/constraints are expressed in another language (e.g. SWRL or SHACL).
3. As an **'OWL2 ontology'**: The [[DPV-OWL]] is a serialisation of the [[DPV]] specification using [[OWL]] language. It should be used where the additional semantic relationships offered by OWL (based on description logic) are needed for modelling knowledge and describing desired inferences. OWL offers more powerful (and complex) features compared to RDFS regarding expression of information and its use to produce desired inferences in a coherent manner.
4. **Creating your own serialisation**: For cases where the above are not suitable or sufficient, an adopter can create their own serialisation of the DPV by implementing the [[DPV]] specification in RDF (or other semantics-aware languages) or for alternate formats and environments such as CSVs, programming APIs, and frameworks. When using DPV in such a manner, it is advised to retain compatibility (and interoperability) by either using the entire IRI (e.g. https://w3.org/ns/dpv#Purpose) or providing documentation for how the custom implementation aligns with the [[DPV]] specification (e.g. stating *MyPurposeConcept* is the same as `dpv:Purpose`). Doing this ensures that the data remains compatible and interoperable with the other uses and applications of DPV.

Areas of Application

The following is an illustrative, but non-exhaustive list of applications possible with the DPV:

- **Document annotation -** identifying and annotating concepts within documents such as privacy policies, legal compliance documentation, web pages
- **Representing Policies** – expressing policies for how personal data should be 'handled', policies for describing an use-cases' use of personal data
- **Representing Rules** – creating and utilising rules for expressing constraints or obligations regarding the use of personal data, checking conformance with obligations such as for legal compliance

For more concrete uses, see the community maintained [[[DPV-ADOPTION]]] in its wiki.

Semantics of DPV

DPV defines a broad notion of semantics for providing a *conceptual* model of concepts and relationships between them. As explained in the [[[#serialisations]]] section earlier, [[DPV]] is the *specification* which is represented formally using [[SKOS]]. To use it as an 'ontology' or 'schema', it is recommended to serialise it into something that can model and represent the required interpretations or constraints. The following sections provides a brief overview of the modelling used in [[DPV]] specification and how it is converted into the ontologies [[DPV-SKOS]] and [[DPV-OWL]].

Concepts and Relationships

[[DPV]] is a collection of *concepts*. Here the term 'concept' is a broadly used as consisting of a *term* non-exhaustively representing any of the following: idea, thought, meaning, object, event, relations, class, or category. Thus, in DPV, 'concepts' consist of terms and relationships between them. These include: *Concept*, *has type*, *is instance of*, and *has applicable concept*.

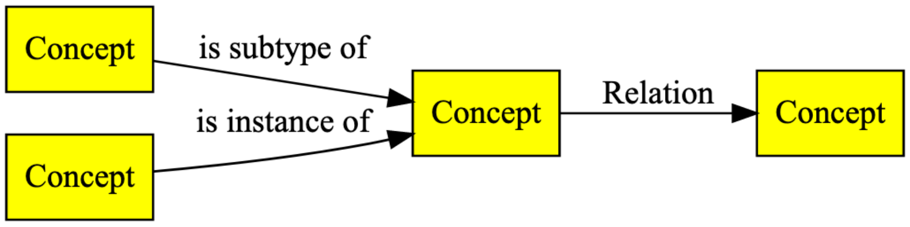

Figure 2: Concepts and Relationships

Consider the example scenario where we want to express the following: Alpha is a *DataController* that *collects Email* as *PersonalData*, *stores* it, and *shares* it with BetaInc (a *Data Processor*), all for *official correspondence*.

In this, the 'basic' or 'core' concepts are: `Data Controller`, `Personal Data`, `Processing`, `Data Processor`, and `Purpose`. Email is another concept that is a specific 'type' or 'subset' or 'category' of Personal Data. Similarly, Processing has 'collects', 'stores', and 'shares'. Alpha is a specific 'instance' of a Data Controller, and similarly BetaInc of a Data Processor.

Here the difference between 'Email' and 'Alpha' is that the former could be further described in terms of the same concept, e.g. Email Address is a specific part of Email as Personal Data; while the latter is 'final' in the sense that it cannot be 'extended' further without losing its meaning. For example, if 'Alpha' has a department or subsidiary, then either of those are not a 'Data Controller' automatically.

In addition to concepts, the above example also requires expressing these *relationships*: (i) a concept is a 'subtype' or 'subset' of another concept; (ii) a concept is an 'instance' of another concept; (iii) indicating a concept is applicable or involved. These relationships are expressed as *is subtype of*, *is instance of*, and *has concept* respectively.

Combining all of these together, we say the following in DPV:

- PersonalData, Purpose, Processing, Data Controller, Data Processor are instances of Concept
- Email is a subtype of Personal Data. Collect, Store, and Share are subtypes of Processing. Official Correspondence is a subtype of Purpose.
- Alpha is an instance of a Data Controller. BetaInc is an instance of a Data Processor.
- The use-case `"has personal data := Email"`; `"has processing := Collect, Store, Share"` ; and so on...
- `has personal data` is a relation linking to `Personal Data`, `has processing` is a relation linking to a `Processing`, and so on ...

A 'concept' in DPV is a 'term' representing information associated with that particular concept. For example, the concept `Email` refers to information about *emails*. This information may contain *email addresses, aliases, signatures*, and so on. While an intuitive use of `Email` may be taken to only refer to *email address*, within DPV concepts are defined with a strict scope as being representatives of all concepts that are inherently a part of it. Therefore, for emails, the concept `Email` is inclusive of email addresses, aliases, and so on from above. To specifically refer to 'email address', the concept `Email Address` should be used, which is a 'subtype' of `Email`.

Through this interpretation, the DPV is structured as a hierarchy of concepts where each *parent or top or broader concept* represents a broad set of information and its *children or bottom or narrower concepts* represent parts of that set.

In the real-world, concepts are not used with the formal and logical consistency expected. Instead, general concepts are used to represent its entire category, which is also why DPV concepts are structured in a top-down fashion. To better reflect a use-case and to ensure accuracy of interpretation when using DPV, it is strongly recommended to either choose concepts which are as detailed or closer to the actual intended usage of that concept, and/or to find a suitable concept and extend it to reflect the specifics as accurately as possible.

In taking this view of concepts and relationships, DPV provides a way to agree upon what a *term* means and is intended to represent. For example, when two different use-cases use the concept *Personal Data* using DPV, both refer to the *same concept*. Similarly, when Email is declared as a subtype of Personal Data, another entity receiving and reading this information must interpret it in the same manner. DPV is thus intended to be a foundational model for terms and relationships when representing and exchanging information.

DPV as an Ontology

Where the [[DPV]] specification defines concepts and relationships between them as 'terms' intended to represent them as 'concepts', the use of such concepts in actual use-cases is often accompanied with additional information and specific 'serialisation' that make it possible to use DPV in a given technological or theoretical framework.

For example, consider the relation `has Personal Data`, which is used to represent association with a `Personal Data` concept or its sub-types or its instances. While this information about what concepts the relationship is being used with/for can be implicit, it can also be explicitly declared as to: (i) express the inherent logic and interpretation explicitly; (ii) provide information for verification of its expression; and (iii) provide hints for identifying concepts to be associated with this relation. For example, specifying that the relation 'has personal data' must always be associated with 'Personal Data'. When considering such uses, DPV must be specified as an 'ontology' using a serialisation that supports representing this and any other required information.

One option to represent ontologies is RDF ([[[RDF]]]) which provides a formal method for expressing information or facts, with RDFS ([[[RDFS]]]) and OWL ([[[OWL]]]) for representing a more detailed and logic-based assertion of the model in terms of relationships and restrictions. While there are other alternatives available to RDF for representing information, and to OWL for representing ontologies, the DPVCG uses these to serialise the DPV specification as an ontology.

The table provides an overview of the expression of concepts across the three DPV serialisations.

Concept

[[DPV]]

[[DPV-SKOS]]

[[DPV-OWL]]

Concept

`dpv:Concept`

`skos:Concept`

`owl:Class`

is subtype of

`dpv:isSubTypeOf`

`skos:broaderTransitive`

`owl:subClassOf`

is instance of

`dpv:isInstanceOf`

`rdf:type`

`rdf:type`

has concept

`dpv:Relation`

`rdf:Property`

`owl:ObjectProperty`

relationship domain

`dpv:domain`

`rdfs:domain`

`rdfs:domain`

relationship range

`dpv:range`

`rdfs:range`

`rdfs:range`

Extending Concepts for Use-Cases

Most of the concepts within DPV are provided as hierarchies of classes representing categories of information, which are generic or abstract or broad so as to permit their application across a diverse and varied landscape of real-world use-cases. In order to accurately reflect the particulars of an use-case, concepts within DPV would (most likely) need to be extended. The specifics for how this should be done depend on the manner in which DPV is utilised. For example, in [[DPV]], the relations `subTypeOf` and `instanceOf` provide a way to indicate such applications.

If using [[DPV-SKOS]] semantics, extending is done using `skos:broaderTransitive`, whereas [[DPV-OWL]] semantics uses the `rdfs:subClassOf` relationship. To create instances, both use `rdf:type`. Where an exact concept is not present within the DPV and a broad concept exists for representing the same information, one should subtype or extend that broad concept to define the required information.

DPV defines the (broad) concept `Marketing` in its `Purpose` hierarchy to represent information about (purposes related to) marketing activities and topics. For a use-case which requires representing purposes (note: plural) related to *marketing of new products*, the broad `Marketing` concept is extended as a *child* or *subclass* concept for representing the intended purpose as, e.g. `MarketingNewProducts`.

Example using [[DPV-SKOS]]

```
# Case1: Where further categories are required to 'group' related purposes
# creating a new subclass or category of Marketing for use-case
ex:MarketingOfNewProducts a dpv:Purpose ;
    skos:broaderTransitive dpv:Marketing ;
    rdfs:label "Marketing of New Products" .

# more specific purposes under group ‘Marketing of New Products’
ex:NewslettersOffers a dpv:Purpose ;
    skos:broaderTransitive ex:MarketingOfNewProducts ;
    rdfs:label "Newsletters about Offers" .
ex:EmailsSeasonalOffers a dpv:Purpose ;
    skos:broaderTransitive ex:MarketingOfNewProducts ;
    rdfs:label "Emails about Seasonal Offers" .

# Case2: A single final and definite purpose within EmailSeasonalOffers
ex:MarketingSeasonalOffer2021 a dpv:Purpose ;
    skos:broaderTransitive dpv:Marketing ;
    rdfs:label "Sending Email Newsletters with Seasonal Offers" .
```

Example using [[DPV-OWL]]

```
# Case1: Where further categories are required to 'group' related purposes
# creating a new subclass or category of Marketing for use-case
ex:MarketingOfNewProducts rdfs:subClassOf dpv:Marketing ;
    rdfs:label "Marketing of New Products" .

# more specific categories of group ‘Marketing of New Products’
ex:NewslettersOffers rdfs:subClassOf ex:MarketingOfNewProducts ;
    rdfs:label "Newsletters about Offers" .
ex:EmailsSeasonalOffers rdfs:subClassOf ex:MarketingOfNewProducts ;
    rdfs:label "Emails about Seasonal Offers" .
```

```
# Case2: A single final and definite purpose within EmailSeasonalOffers
ex:MarketingSeasonalOffer rdf:type dpv:Marketing ;
    rdfs:label "Sending Email Newsletters with Seasonal Offers" .
```

The mechanism for extending concepts (via both subclasses/subtypes and instances) is useful to align existing concepts or vocabularies with the DPV taxonomies, such as by declaring them as subclasses of a particular concept. This permits the creation of domain or jurisdiction specific *extensions*, such as [[DPV-GDPR]] for expressing the legal bases provided by GDPR. Extensions also permit more accurate representations of a use-case by extending from multiple concepts to refine and scope the interpretation. This means each concept can have multiple parents representing the intersection of their respective sets.

Consider the use-case where an activity simultaneously uses the data while collecting it. The first representation (`ActivityA`) models them separately - which is not accurate as it is ambiguous in terms of collection and usage taking place independently. By extending the collect and use concepts to create a new concept called `CollectAndUse`, it is possible to accurately declare that they both occur as a concurrent operation. Such combinations of concepts are also useful to collectively represent or annotate additional information such as: technologies involved, implementation details, or agents involved.

Example in [[DPV-SKOS]]

```
# Method 1: Ambiguous regarding independence of Collect and Use
ex:ActivityA a dpv:PersonalDataHandling ;
    dpv:hasProcessing ex:Collect, ex:Use .

# Method 2: Accuracy regarding combination of Collect and Use
ex:CollectAndUse skos:broaderTransitive ex:Collect, ex:Use ;
    rdfs:label "Collect and Use data using User Device" .
ex:Activity a dpv:PersonalDataHandling ;
    dpv:hasProcessing ex:CollectAndUse .
```

Example in [[DPV-OWL]]

```
TODO
```

It is not necessary to extend concepts unless one wishes to depict use-case specific information. For example, if in a use-case it is sufficient to (only) say some information is collected, then `dpv:Collect` can be directly used. However, where more specific information is needed, such as also specifying a method of collection (e.g. `CollectViaWebForm`), then it is recommended to extend the concept, for example as `<CollectViaWebForm a dpv:Collect>`. If there are lots of forms and they need to be 'grouped' together as collection methods, then one would subtype/subclass `Collect` as `CollectViaWebForm` and create instances of it for each form to be represented.

Though this example used a web form as a method of collection by directly mentioning it within the concept as `CollectViaWebForm`, this may not always be desirable. For example, that same web form may also need to be represented separately for logging purposes. DPV is exploring the provision of a `Technology` concept to assist in representing information regarding how concepts are implemented and the use of specific technological artefacts such as web forms, databases, along with their functions such as data storage and retrieval.

Maintaining Interoperability

DPV intends to provide a base or foundational framework for different entities to exchange information and interpret concepts for interoperability. When an adopter (e.g. an organisation using DPV) extends concepts to refine them for their own use-case, the concept is still (weakly) interoperable by relying on DPV's broad taxonomies to provide a common point of reference.

For example, two TV companies (`AliceCo` and `BobCo`) extend the concept `Optimisation` to reflect their respective purposes as follows:

Example in [[DPV-SKOS]]

```
# AliceCo’s optimisation related to better services for users’ infrastructure
exA:TVServiceOptimisaion skos:broaderTransitive dpv:OptimisationForConsumer ;
    rdfs:label "Optimise Service for Users’ Infrastructure" .

# BobCo’s optimisation related to more efficient signals for users’ TV sets
exB:TVSignalOptimisation skos:broaderTransitive dpv:OptimisationForConsumer ;
    rdfs:label "Optimise Signal for Consumer TV Set" .
```

When exchanging information about their use-cases with each other (or with a third party), by following the chain of use-case specific concepts it is possible to deduce that both `AliceCo` and `BobCo` are doing optimisations for consumers. Thus a common language or interface can be developed based on using DPV as a point of interoperability and commonality which can be used by adopters to define the specifics of their use-case. For example, in the above use-case, a common notice generation algorithm could be created and used to inform users of both services the purposes each company is using data for.

```
# analysing respective graphs, a common ancestor is found as:
# dpv:OptimisationForConsumer ; Using this as context to compare:
# (either manually, or based on data used, etc.)

# 1: BobCo's optimisations are found to be broader than AliceCo's
exA:TVServiceOptimisation skos:broaderTransitive exB:TVServiceOptimisation .

# 2: BobCo's optimisations are found to be the same as AliceCo's
exA:TVServiceOptimisation skos:exactMatch exB:TVServiceOptimisation .

# 3: BobCo's optimisations are found to be similar to AliceCo's
```

```
exA:TVServiceOptimisation skos:closeMatch exB:TVServiceOptimisation .
```

Core Concepts

Structure of DPV

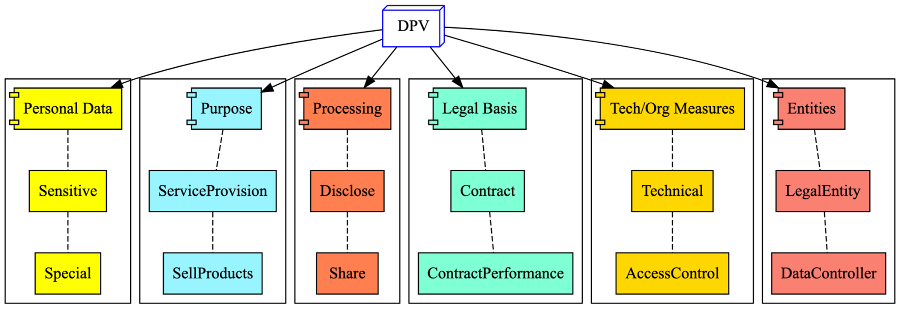

Figure 3: DPV as a set of hierarchies

DPV can be viewed as a hierarchical taxonomy of concepts where each `core concept` represents the top-most abstract concept in a tree and each of its children provide a lesser abstract or more concrete concept. For example, consider the concept of `PersonalData` which is the abstract representation of *personal data*. It can be further *refined* or *extended* as `SensitivePersonalData`, and further as `SpecialCategoryPersonalData` and then as `GeneticData` and so on.

From this perspective, the top-most abstract concepts are collectively referred to as the *core vocabulary* within DPV. The goal of the DPV is to provide a rich collection of concepts for each of top concepts so as to enable their application within real-world use-cases. The identification of what constitutes a core concept is based on the need to represent information about it in a modular and independent form, such as that required for legal compliance.

Each core concept is intended to be independent from other core concepts. For example, the `Purpose` (e.g. Optimisation) refers only to the *purpose of why personal data is processed* and is independent as a concept from the `PersonalData` (e.g. `Location`) or the `Processing` activities (e.g. `collect`, `store`) involved to carry out that purpose. Such separation is necessary in order to represent and answer questions such as:

- *Q:* What data is being processed?
  Ans: `dpv:PersonalData` → `dpv:Email`
- *Q:* Why is the data being processed?
  Ans: `dpv:Purpose` → `dpv:Marketing`
- *Q:* What is being done with the data?
  Ans: `dpv:Processing` → `dpv:Collect`, `dpv:Store`

The separation of concepts creates a *modular* structure for concept hierarchies within DPV, which in turn allows an adopter to use one particular concept taxonomy or module (e.g. list of purposes) independently without reusing the others, or to select only those concepts which are needed for their particular use-case. The separation also permits greater flexibility of representation and usage - such as using different combinations of core concepts as needed in use-cases. For example, a use-case can specify a single concept represent both Purpose and Processing by combining their respective concepts from DPV. The modular design of DPV also makes it possible to define domain and jurisdiction specific concepts in a separate namespace - such as the [[[DPV-NACE]]] purpose taxonomy providing a way for `Purpose` to indicate sectors using NACE taxonomy, and the [[[DPV-GDPR]]] for using `LegalBasis` to represent the legal bases provided by GDPR.

Overview of Core Concepts

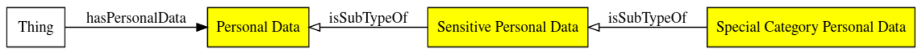

Figure 4: Indicating applicable or relevant `PersonalData`

PersonalData ‘Personal data’ refers to any data about a natural person that can be used to identify them directly or, in combination with other information, indirectly. ‘Personal data’ is also commonly referred to as ‘personally identifiable information (PII)’. However the terms should not be interchangeably used as based on definitions (e.g. those in GDPR), ‘personal data’ can be interpreted as a broader term than PII, and where PII may refer to <u>only</u> to information that can directly identify a person. DPV’s definition of personal data is based on the broadest possible definition (i.e. from GDPR) as it covers a wider range of information considered ‘personal data’. Personal data can be declared as a category, such as ‘Email’, or an instance, such as ‘x@y.z’. `PersonalData` is associated to using the relation `hasPersonalData`.

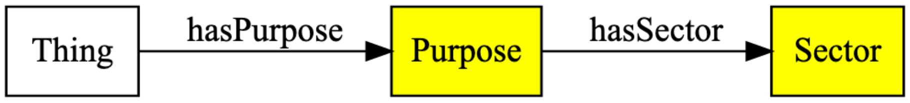

Figure 5: Indicating applicable or relevant `Purpose`

Purpose Representing the *purpose* for which personal data is processed, for e.g. ‘Personalisation’ as a broad category of purpose. Information about the purpose can be further specified by denoting information about its interpretation within a particular `Sector`, such as from standardised authoritative lists e.g. [[NACE]].

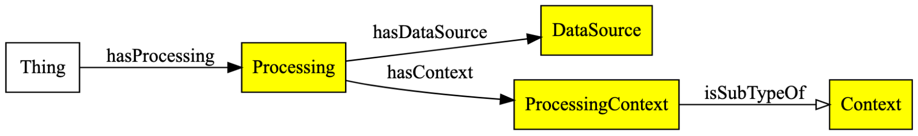

Figure 6: Indicating applicable or relevant `Processing`

Processing Representing *processing* as in the actions or operations over personal data, for e.g. collect, use, share, store. To indicate the origin or source of data, the concept `DataSource` along with relation `hasDataSource` is provided. For additional *contextual* information regarding operations or processing, such as whether it include humans or automation, the concept `ProcessingContext` is provided which can be associated using the relation `hasContext` (description of `Context` is provided later in the document). Examples of *ProcessingContext* include conditions such as profiling, automated decision making, human involvement.

Thing
hasLegalBasis
LegalBasis

Figure 7: Indicating applicable or relevant `LegalBasis`

LegalBasis A legal basis is a law or a clause in a law that justifies or permits the processing of personal data in the specified manner. It is a jurisdictional concept given the scoping of laws to specified countries or regions, as well as a domain-specific concept given the specific laws enacted scoped to particular domains. A law, such as the GDPR, that regulates the use of personal data requires that every processing of personal data must be justified with some legal basis to ensure it is lawful, and to further assess its correctness, accountability, and impact based on the obligations applicable. However, what is considered a legal basis varies greatly across cultures, domains, use-cases, and laws themselves. The aim of DPV is therefore to provide an upper-level abstract taxonomy of categories of legal bases that can be customised and applied as needed.

Entities Representing the 'entities' or 'actors' involved in the processing of personal data. DPV provides a broad categorisation of entities based on their relevance in jurisprudence (i.e. legal roles) as well as categorisation in real-world (e.g. organisation types).

DataController Representing the organisation(s) responsible for processing the personal data.

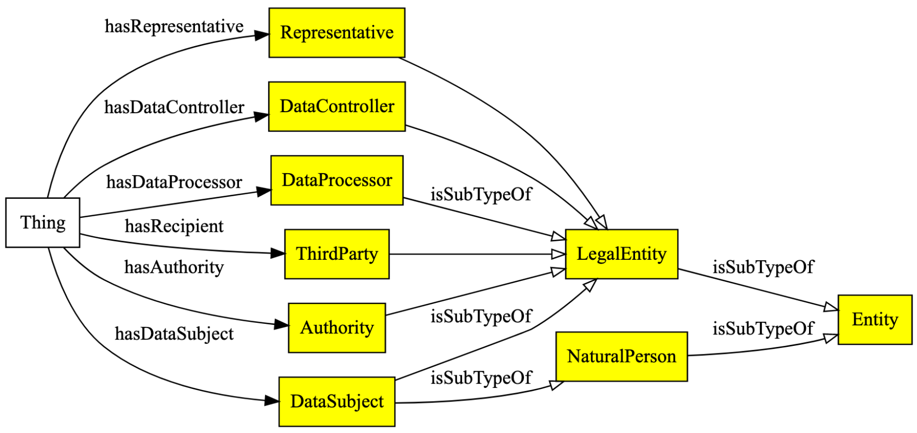

Figure 8: Indicating applicable or relevant `Entities`

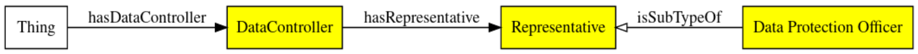

Figure 9: Indicating applicable or relevant `DataController`

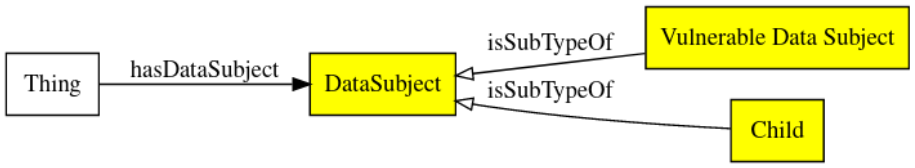

Figure 10: Indicating applicable or relevant `DataSubject`

DataSubject Representing the categories or groups (e.g. Users of a Service), or instances (e.g. Jane Doe) of individual(s) whose personal data is being processed.

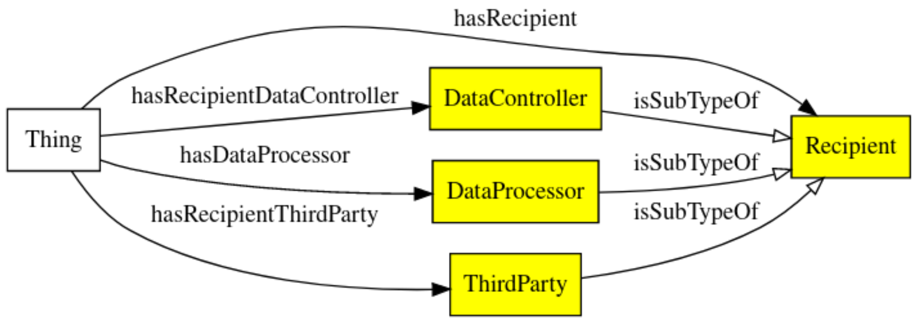

Figure 11: Indicating applicable or relevant `Recipient`

Recipient Represents the entities that receive personal data, e.g. when it is shared.

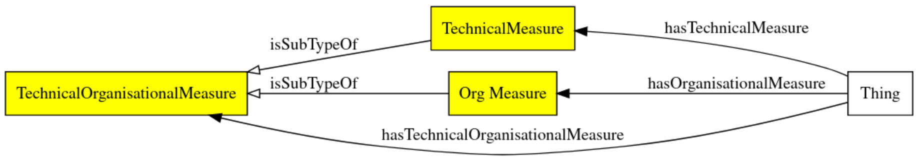

Figure 12: Indicating applicable or relevant `TechnicalOrganisationalMeasure`

TechnicalOrganisationalMeasure DPV provides a taxonomy of technical and organisational measures for representing information about how the processing of personal data is technically and organisationally protected, safeguarded, secured, or otherwise managed. This is distinct from what technology is used for carrying out processing, and instead refers to what *measures* are in place (i.e. what the technology intends to provide in terms of features).

Technical and Organisational measures consist of activities, processes, or procedures used in connection with ensuring data protection, carrying out processing in a secure manner, and complying with legal obligations. Such measures are required by regulations depending on the context of processing involving personal data. For example, GDPR (Article 32) states implementing appropriate measures by taking into account the state of the art, the costs of implementation and the nature, scope, context and purposes of processing, as well as risks, rights and freedoms.

The broad concept `TechnicalOrganisationalMeasure` represents all technical and organisational measures, which are associated through the relation `hasTechnicalOrganisationalMeasure`. The concept `TechnicalMeasure`, associated using the relation `hasTechnicalMeasure`, concerns measures primarily achieved using some technology. Similarly, `OrganisationalMeasure` and the relation `hasOrganisationalMeasure` represent measures carried out through activities and processes at the management and organisational levels, which may or may not be assisted by technology.

Specific examples of measures in the article include:

- the pseudo-anonymisation and encryption of personal data;
- the ability to ensure the ongoing confidentiality, integrity, availability and resilience of processing systems and services;
- the ability to restore the availability and access to personal data in a timely manner in the event of a physical or technical incident;
- a process for regularly testing, assessing and evaluating the effectiveness of technical and organisational measures for ensuring the security of the processing.

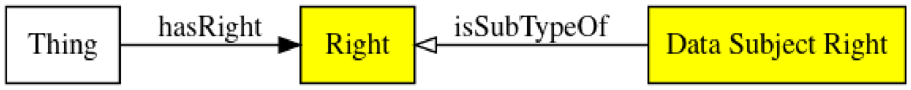

Figure 13: Indicating applicable or relevant `Right`

Right Representing the rights available, applicable, or afforded by a law or regulation, either to data subjects or data controllers, or other entities.

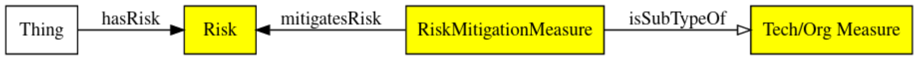

Figure 14: Indicating applicable or relevant `Risk`

Risk Representing risk(s) associated with a concept, for e.g. risk of unauthorised data disclosure related to processing, technical measure, or vulnerability of data subjects

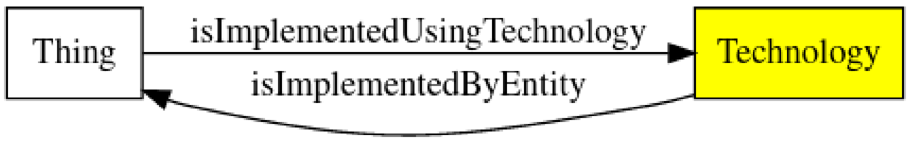

Figure 15: Indicating applicable or relevant `Technology`

Technology Representing the rights available, applicable, or afforded by a law or regulation, either to data subjects or data controllers, or other entities

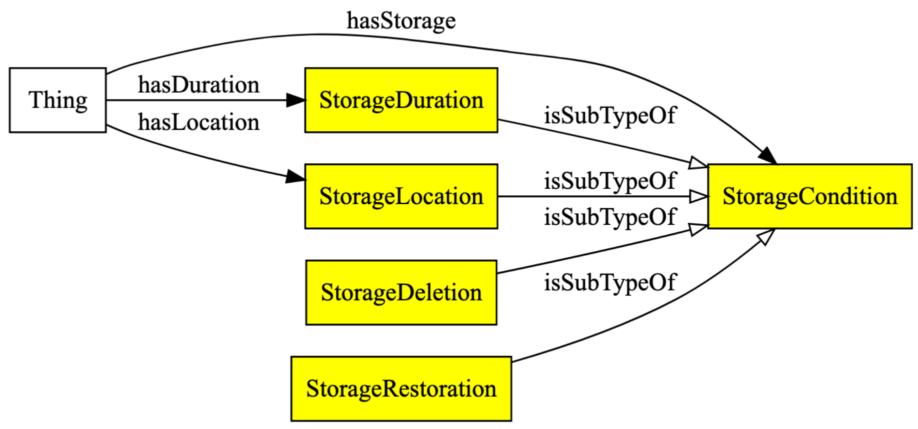

Figure 16: Indicating applicable or relevant temporal and geo-spatial information

Temporal and Geo-Spatial Information for Storage Indicating information about storage of personal data, such as its location, duration, deletion (e.g. erasure mechanisms), or restoration (e.g. backup availability). Storage information can be part of the *processing* information (e.g. logs) or *technical and organisational measure* (e.g. indicating policies or plans in place) depending on context.

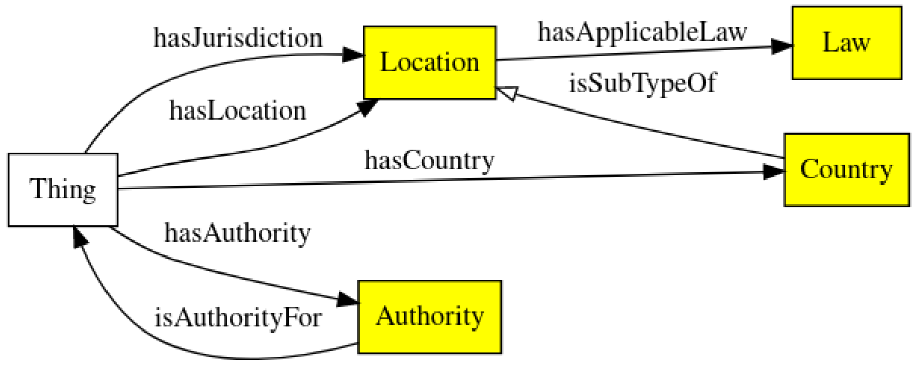

Figure 17: Indicating applicable or relevant `Jurisdiction`

Location and Jurisdiction Representing the rights available, applicable, or afforded by a law or regulation, either to data subjects or data controllers, or other entities

Personal Data Handling

In legal terminology, it is common to refer to all information about how personal data is being processed using the colloquial term *processing*. This results in confusion between the use of *processing* as a concept referring to all information (i.e. purposes, personal data, collection, storage, etc.), and `processing` as a concept referring to (only) the specific actions or operations (e.g. collect, use).

To avoid this ambiguity and enable clarity of information, DPV defines a new concept called `PersonalDataHandling` for representing how the core concepts are related or apply to each other for a particular use-case. The association of a concept to `PersonalDataHandling` is made using the relationships or properties provided for each concept. For example, to indicate a `PersonalDataHandling` includes personal data, the relationship `hasPersonalData` is used along with the concept `PersonalData`. The following figure provides an overview of how the `PersonalDataHandling` concept provides a way to associate relevant concepts with one another through it.

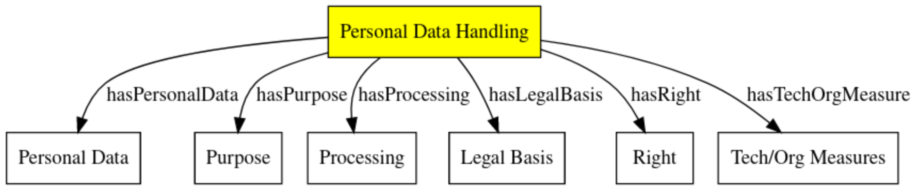

Figure 18: `PersonalDataHandling` as a central concept

For an example of how `PersonalDataHandling` brings together the core concepts, consider the example where `Acme` is a `DataController` that `Collect`(s) and `Use`(s) `Email` for `ServiceProvision`.

Example using [[DPV-SKOS]]

```
ex:Acme rdf:type dpv:DataController .
ex:AcmeMarketing rdf:type dpv:PersonalDataHandling ;
    dpv:hasDataController ex:Acme .
    dpv:hasPersonalDataCategory dpv:EmailAddress ;
    dpv:hasProcessing dpv:Collect, dpv:Use ;
    dpv:hasPurpose dpv:ServiceProvision ;
```

Example using [[DPV-SKOS]]

```
TODO
```

Note that `PersonalDataHandling` is intended to provide a convenient concept for tying the core concepts together, and DPV does not make its use binding, nor does it constrain the relationships to only be defined between `PersonalDataHandling` and the other core concepts. This is so as to permit using DPV in alternate or differing models. For example, where a central concept already exists, such as when describing relevant information for a smartphone app, the concept for `App` can be a replacement for `PersonalDataHandling` based on statements such as `<App> hasPurpose <SomePurpose>`. Even in such cases, `PersonalDataHandling` can provide granular expression thereby enabling description of different contexts within which the app uses personal data, such as for registration or complaint resolution.

Nesting `PersonalDataHandling` to express granular models The use of `PersonalDataHandling` can be nested, which means one instance can contain other instances, much like a box with several smaller boxes inside. This permits breaking down complex or dense use-cases into more granular ones and representing them in a more precise and granular fashion. In the above example, consider the following situation containing a single `PersonalDataHandling` instance comprising of two additional instances representing: (i) data is stored using a data processor, (ii) data is used for Marketing. While it is certainly possible to represent all of this information within one single instance of `PersonalDataHandling`, the adopter may decide to create separate instances of `PersonalDataHandling` based on requirements such as reflecting similar separations for legal documentation or accountability purposes.

Consider the example where *Acme*, as a *DataController*, maintains records of its processing activities using `PersonalDataHandling` to represent one of its services. In this, it collects email, uses it for internal analyses based on *LegitimateInterests*, and also sends marketing information by using a processor based on the data subject's consent. Using nesting of personal data handling, the information can be expressed at granular level representing service, individual purposes, and so on.

```
ex:Acme rdf:type dpv:DataController .
ex:AcmeMarketing rdf:type dpv:PersonalDataHandling ;
    dpv:hasPersonalDataHandling ex:InternalAnalytics ;
    dpv:hasPersonalDataHandling ex:SendingNewsletters .

ex:InternalAnalytics rdf:type dpv:PersonalDataHandling ;
    dpv:hasPersonalData dpv:Email ;
    dpv:hasProcessing dpv:Collect, dpv:Store ;
    dpv:hasPurpose dpv:InternalResourceOptimisation ;
    dpv:hasLegalBasis dpv:LegitimateInterestOfController .

ex:FooTech rdf:type dpv:DataProcessor .
ex:SendingNewsletters rdf:type dpv:PersonalDataHandling ;
    dpv:hasPersonalData dpv:Email ;
    dpv:hasProcessing dpv:Share ;
    dpv:hasPurpose dpv:Marketing ;
    dpv:hasDataProcessor ex:FooTech ;
    dpv:hasLegalBasis dpv:Consent .
```

Alternate Models to PersonalDataHandling An instance where one may not wish to utilise `PersonalDataHandling` is where the adopter or use-case wants to indicate a different method for relating concepts together. For example, instead of expressing the relationship between personal data and purpose through a `PersonalDataHandling` instance, an alternate model could be one where the purpose directly specifies what personal data it uses as: `<SomePurpose hasPersonalData SomePersonalData>`. Similarly, another instance for such alternate use of concepts is to associate a legal basis directly with the purpose by using the `hasLegalBasis` relationship. To support such uses, DPV does not explicitly declare restrictions on the properties in terms of what concepts they can be used with (e.g it does not provide domain assertions). In case an adopter needs such explicit declarations, they can utilise or import the separate file declaring them.

The following figure indicates an alternate model which does not use `PersonalDataHandling` as a central concept, but instead uses the core concepts and relationships to structure information related to a *Service*.

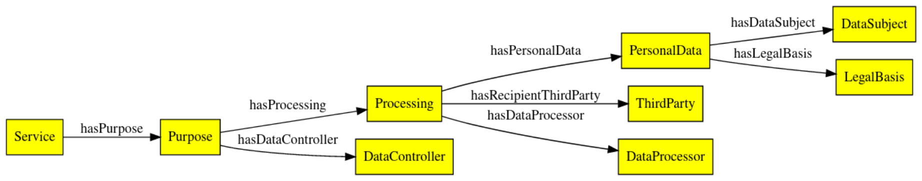

Figure 19: Alternate model to `PersonalDataHandling` using core concepts and relationships

When using custom-defined restrictions and data models, it is important to note the consequences such models have on interpretation and interoperability of data defined using DPV. For example, consider a compliance assessment tool that takes DPV data as input. If the tool expects a `PersonalDataHandling` with links to relevant information, using other alternate models and relationships can produce invalid or incorrect results. To avoid this, we recommend:

1. Documenting alternate models to clearly indicate their interpretation and use of DPV semantics;
2. Where possible, ensuring and providing mappings between the alternate models and the `PersonalDataHandling` or equivalent concepts within DPV so that the data can be transformed for interoperability;
3. Consider contributing your idea or implementation of an alternate model to DPVCG to create a 'library of models', which can act as documentation for adopters and provide better understanding of the models impacts on requirements and interpretation of information specified using DPV. This

exercise can also assist in selecting a common model as the 'default' and to provide mechanisms for conversion/interoperability between it and other models.

Taxonomies of Key Concepts

The following sections provide an overview of the taxonomies (i.e. hierarchies of concepts) provided by DPV for its core concepts.

Purpose

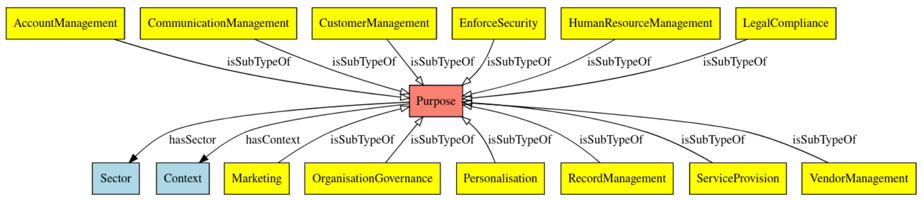

Figure 20: Overview of top-level concepts in `Purpose` taxonomy

DPV's taxonomy of purposes is used to represent the *reason* or *justification* for processing of personal data. For this, purposes are organised within DPV based on how they relate to the processing of personal data in terms of several factors, such as: management functions related to information (e.g. records, account, finance), fulfilment of objectives (e.g. delivery of goods), providing goods and services (e.g. service provision), intended benefits (e.g. optimisations for service provider or consumer), and legal compliance.

It is important to note the following for real-world implications of `Purpose`:

1. There is no universal definition for what constitutes a 'purpose' or what attributes are associated with it.
2. There are several distinct ways to model purposes, e.g. as a 'goal' such as 'Delivery of Ordered Goods'; or as a statement explaining the processing of personal data, e.g. 'Sending newsletters to Email'.
3. DPV does not define requirements for what is a 'valid purpose' as these are defined externally, e.g. in laws such as [[GDPR]] Article.5-1b where purposes are required to be 'explicit and legitimate'.
4. Purposes have contextual interpretations within their application and domains i.e. depending on how they are used in an use-case). For example, `ServiceProvision` is interpreted distinctly across the use-cases of an online website, a goods delivery outlet, and a medical centre - even if they use the same term or wording.

Following from the above, most use-cases would need to extend one of the concepts within DPV's purpose taxonomy to ensure its purpose descriptions are specific and understandable within the context of that use-case. We therefore suggest, where possible and appropriate, to create a customised purpose as required within the use-cases by extending or subtyping one or several purposes from the DPV taxonomy and to provide a human readable description to assist in its accurate interpretation (e.g. for RDF, using `rdfs:label` and `rdfs:comment`).

In this example, a new purpose is created by extending `dpv:FraudPreventionAndDetection` and annotated with human-readable information. The interpretation of this purpose is thus more clear in relation to how it is applied or used within that use-case, and also serves to compare it with other purposes within the same category.

Example using [[DPV-SKOS]]

```
ex:PreventTransactionFraud a dpv:Purpose ;
  skos:broaderTransitive dpv:FraudPreventionAndDetection ;
  skos:prefLabel "Prevent Transaction and Payment Frauds" ;
```

Sector of Purpose Application DPV provides `Sector` that can be used to indicate the relevant information to further clarify or indicate how a purpose should be interpreted. `Sector`, used with the `hasSector` relation, denotes the *sector* or *domain* of application, such as Manufacturing. This can be used alongside existing official sector taxonomies such as [[NACE]] (EU), [[NAICS]] (USA), or [[ISIC]] (UN), as well as commercial industry taxonomies such as [[GICS]] maintained by organisations MSCI and S&P. Multiple classifications can be used through mappings between sector codes such as the [[[NACE-NAICS]]] provided by EU.

DPVCG provides an interpretation of the NACE revision 2 codes which uses `rdfs:subClassOf` to specify the hierarchy between sector concepts. It is available as [[DPV-NACE]]. The NACE codes within this extension have the namespace `dpv-nace` and are represented as `dpv-nace:NACE-CODE`.

We are working on further alignments between the NACE codes and DPV's purpose taxonomy, and welcome contributions for the same.

For example, the following purpose concerns implementing access control with the domain specified as scientific research using its corresponding NACE code `M72` to indicate sectorial implications for what "access control" and "enforce security" are expected to imply.

Example using [[DPV-SKOS]]

```
:RestrictPersonnelAccess a dpv:Purpose ;
  skos:broaderTransitive dpv:EnforceSecurity ;
  skos:prefLabel "Limit access to lab by checking personnel identity" ;
  dpv:hasSector dpv-nace:M72 .
```

While the use of `Sector` for restricting (personal data processing) purposes is an uncommon and undocumented practice in terms of legal enforcement, we provide this feature as the use of sector code can assist with identification and interpretation of information as well as legal or organisational obligations and policies. For example, indicating some purpose is to be implemented within manufacturing or scientific research facilities (e.g. medical centres) can assist in ensuring specific types of access control and policies are defined and implemented.

Processing Categories

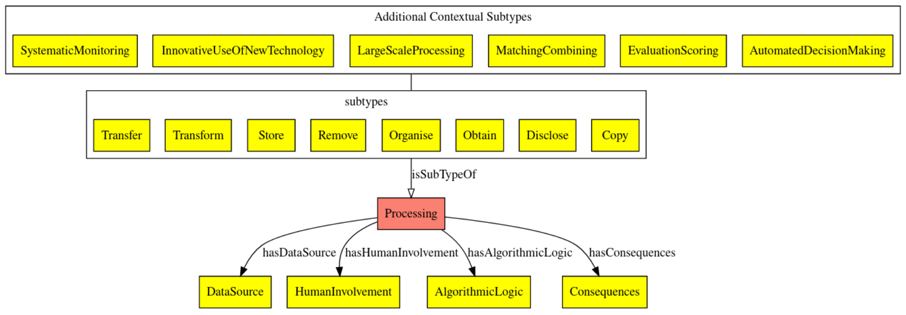

Figure 21: Overview of top-concepts in `Processing` taxonomy

DPV's taxonomy of processing concepts reflects the variety of terms used to denote processing activities or operations involving personal data, such as those from [[GDPR]] Article.4-2 definition of *processing*. Real-world use of terms associated with processing rarely uses this same wording or terms, except in cases of specific domains and in legal documentation. On the other hand, common terms associated with processing are generally restricted to: collect, use, store, share, and delete.

DPV provides a taxonomy that aligns both the legal terminologies such as those defined by GDPR with those commonly used. For this, concepts are organised based on whether they subsume other concepts, e.g. `Use` is a broad concept indicating data is used, which DPV extends to define specific processing concepts for `Analyse`, `Consult`, `Profiling`, and `Retrieving`. Through this mechanism, whenever an use-case indicates it *consults* some data, it can be inferred that it also *uses* that data.

The definitions for describing and interpreting each processing concept is based on the following sources: language dictionaries (pre-dominantly Oxford English), use of the term within legal documents (e.g. GDPR case law), and technology-specific interpretations such as for IT systems. Despite these, there may be distinct interpretations for what a term represents based on differences in practices, culture, language, and domains. In case an adopter or a use-case foresees such ambiguity or confusion, it is advisable to extend the relevant concepts and define them as needed, or create a separate extension.

Indicating Processing Location and Duration For indicating details of processing, such as where and when it is taking place in terms of location or frequency, the concepts and relationships associated with these concepts can be used. Where possible, it is recommended to utilise common vocabularies and methods of specifications to promote interoperability of information, such as [[TIME]] for temporal information.

In this example, Acme is a Data Controller whose servers are located in Ireland, and which processes data for a period of one year

```
ex:ServerInfo a dpv:PersonalDataHandling ;
    dpv:hasDataProcessor ex:Acme ;
    dpv:hasProcessing ex:IEServers .

ex:IEServers a dpv:Processing ;
    dpv:broaderTransitive dpv:Use ;
    dpv:hasLocation ex:IE ;  # IE is ISO 3166-1 alpha-2 country-code for Ireland
    dpv:hasDuration [
        rdf:type time:Duration, dpv:Duration ;
        time:numericDuration "1"^^xsd:decimal ;
        time:unitType time:unitYear ;
      ] .
```

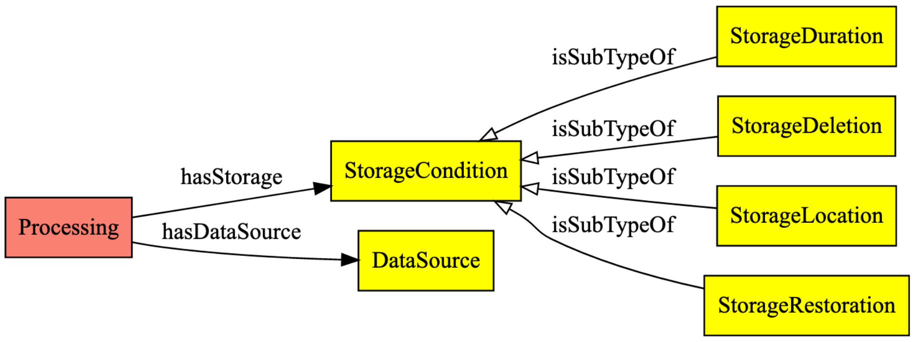

Figure 22: Indicating Storage and Data Source for Processing

Indicating Storage and Source of Data The DPV processing taxonomy provides the concept `Store` to indicate data is being stored. To specify additionally information, such as the location of where the data is being stored, or how it will be deleted, generic properties associated with processing (i.e. *location* and *duration*) can be used. However, given the importance of information related to data storage in terms of use-cases, especially policies and legal compliance, DPV provides additional concepts and relationships under the general concept `StorageCondition` and the relation `hasStorage`.

For declaring the source of data, the `DataSource` concept along with `hasDataSource` relationship is provided to indicate where the data is collected or acquired from. For example, data can be obtained from the data subject directly (e.g. given via forms) or indirectly (e.g observed from activity, or inferred from existing data), or from another entity such as a third party.

This example is similar to the previous one, where Acme is a Data Processor that collects data from BetaInc, a Data Controller, and stores it within servers located in Ireland for a period of one year.

```
ex:ServerInfo a dpv:PersonalDataHandling ;
    dpv:hasDataProcessor ex:Acme ;
    dpv:hasProcessing ex:DataFromBeta, dpv:Use ;

ex:DataFromBeta a dpv:Processing ;
    dpv:broaderTransitive dpv:Collect, dpv:Store ;
    dpv:hasLocation ex:IE ;
    dpv:hasDataSource ex:BetaInc ;
    dpv:hasDuration [
        rdf:type time:Duration, dpv:Duration ;
        time:numericDuration "1"^^xsd:decimal ;
        time:unitType time:unitYear ;
      ] .
```

Indicating Scale, Automation, and New (Untested) Technologies For indicating information relevant to context and impact of processing, DPV provides concepts for representing whether processing is carried out at large scale, consists of systematic monitoring (of data subjects), is performed for evaluation or scoring (of data subjects), matching and/or combining existing datasets, utilizes automated decision making, or involves innovative use of new technologies. These concepts are based on [[GDPR]] Article.35 regarding Data Protection Impact Assessments (DPIA), and provide a way to indicate their applicability for specific processing activities - such as specifying collection happens at large scale or that data is used for/with automated decision making.

Consider the use of a spam filter that is based on automated processing operations that collects information at large scales, analyses it against some novel spam detection criteria, and makes automated decisions whether to permit communication to proceed.

```
ex:SpamFilter rdf:type dpv:PersonalDataHandling ;
    dpv:hasProcessing ex:DataCollection, ex:SpamDetection .

ex:DataCollection rdf:type dpv:Processing ;
    skos:broaderTransitive dpv:Collect ;
    dpv:hasProcessingContext dpv:LargeScaleProcessing .
```

```
ex:SpamDetection rdf:type dpv:Processing ;
    skos:broaderTransitive dpv:Analyse ;
    dpv:hasProcessingContext dpv:AutomatedDecisionMaking,
                             dpv:InnovativeUseOfNewTechnologies .
```

We invite additional contributions regarding concepts and relationships for representing contextual information, such as the scale of processing, or factors involved in decision making.

Personal Data

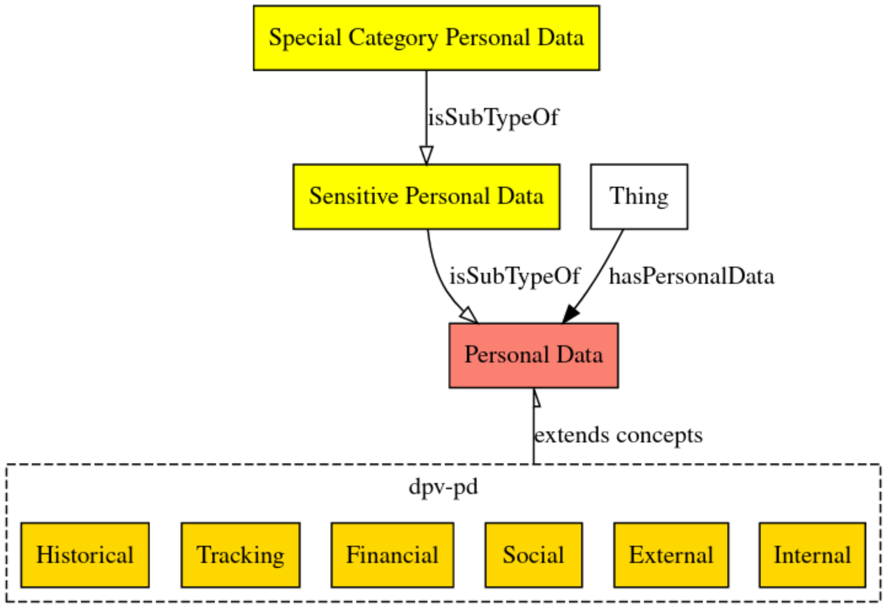

Figure 23: Personal Data concepts within DPV and their extension in `dpv-pd`

DPV provides the concept `PersonalData` and the relation `hasPersonalData` to indicate what categories or instances of personal data are being processed. As described earlier, common use of personal data concepts in the real-world consists of specifying as concepts both categories (e.g. *Location*) and instances (e.g. your *exact location right now*).

Real-world and common usage of *personal data* is at both an abstract level as well as specific level. For example, consider the sentence "We use your Email information...", which uses "*Email*" to represent a reference to what personal data is involved. Here, one may interpret *Email* as representing only the *email address*, or as a broad set of possible information related to emails, such as *email* *address, email senders and recipients list, email service provider, email usage statistics* and so on.

For ensuring clarity and resolving any potential ambiguity, DPV recommends being as specific as possible. This means where there is ambiguity as to what the information may be associated with or within a concept, it is advisable to resolve that ambiguity - either by choosing a more accurate concept from the taxonomy and/or by creating one through extension of an existing concept.

In addition to above, it is also challenging to accurately represent how concepts function within real-world use in terms of their encapsulation within one another. For example, when establishing the DPV, we discussed the modelling of personal data categories based on the scenario where a picture of passport is initially collected, and from it various categories are extracted, such as - name, address, and photo. For representing this, merely stating the personal data as 'passport photograph' would not be entirely accurate as there is additional information within the photograph.

A solution was established whereby the use-case is expected to declare what information it intends to collect or use through the mechanism of expression relation between its personal data categories. For the passport photograph scenario, the use-case would declare the class `PassportPhoto` with subtypes representing `Name`, `Age`, and so on. This is necessary to ensure the interpretation that using `PassportPhoto` means having access to and using all of its subsequent personal data categories.

While this is one possible solution, other methods exist, such as explicitly declaring the data categories and their encapsulation within one another, such as by reusing `hasPersonalData` or creating additional properties (e.g. `containsData`) to indicate a personal data concept, i.e. the passport photo, *contains* information associated through the relation, i.e. name, age, etc. We welcome discussions regarding both these methods.

`dpv-pd` extension The [[DPV-PD]] extension provides additional concepts that extend the DPV's personal data taxonomy based on an opinionated structure contributed by R. Jason Cronk from EnterPrivacy. This separation is to enable adopters to decide whether the extension's concepts are useful to them, or to use other external vocabularies, or define their own.

Concepts within [[DPV-PD]] are broadly structured in top-down fashion by utilising their relevance and origin as:

- Internal (within the person): e.g. Preferences, Knowledge, Beliefs
- External (visible to others): e.g. Behavioural, Demographics, Physical, Sexual, Identifying
- Social: e.g. Family, Friends, Professional, Public Life, Communication
- Financial: e.g. Transactional, Ownership, Financial Account
- Tracking: e.g. Location, Device based, Contact

- Historical: e.g. Life History

Sensitive and Special Categories For indicating personal data which is sensitive, the concept `SensitivePersonalData` is provided. For indicating special categories of data, the concept `SpecialCategoryPersonalData` is provided. In this, the concept *sensitive* indicates that the data needs additional considerations (and perhaps caution) when processing, such as by increasing its security, reducing usage, or performing impact assessments. *Special categories*, by contrast, are a 'special' type of sensitive personal data requiring additional considerations or obligations defined in laws that regulate how they are used or prohibit their use until specified obligations are met. DPV currently adopts the special categories defined in [[GDPR]] Article 9.

The sensitivity of personal data can be *universal*, where that data is always sensitive, or *contextual*, which means a use-case needs to declare it as such. For indicating personal data used is sensitive (or special), it is sub-typed or declared as an instance of `SensitivePersonalData`, as shown in the example below.

In using these concepts, it is important to note that **DPV's modelling of sensitive and special categories is non-exhaustive** and as such should not be taken as the only source of truth. To assist with better identifying sensitive concepts, work is ongoing within DPV to identify and provide a reference list of sensitive and special categories, and we welcome contributions for the same.

In a medical study involving genetic data, the person's collected blood samples are considered to be 'special category' personal data under the GDPR. In addition to this, the person's identifier associated with that dataset to enable monitoring future appointments is considered sensitive within the medical centre. Note that the examples uses the [[DPV-PD]] extension to represent some concepts.

In this example, the knowledge that blood samples are of type 'special category' can be inferred from the fact that they are a form of *Medical Health* which is a 'special category'. However, the example considers best practices that suggest explicitly identifying an denoting that blood samples are also of type 'special category'.

```
ex:PatientStudy rdf:type dpv:PersonalDataHandling ;
    dpv:hasPersonalData ex:BloodSamples, ex:PatientIdentifier .

ex:BloodSamples rdf:type dpv:SpecialCategoryPersonalData ;
    skos:broaderTransitive dpv-pd:MedicalHealth ;
    skos:narrowerTransitive dpv-pd:BloodType .

ex:PatientIdentifier rdf:type dpv:SensitivePersonalData ;
    skos:broaderTransitive dpv-pd:Identifying .
```

Derived and Inferred Data For indicating personal data is derived or inferred from other personal data, the concepts `DerivedPersonalData` and `InferredPersonalData` are provided. In DPV, the method for derivation or inference is considered to be part of the `Processing` concept and therefore should be declared using the appropriate processing concepts. However, an adopter can choose an alternate model for declaring information about how data was derived or inferred, such as by using PROV to model data as inputs or outputs of activities.

This use-case collects browser fingerprint and IP Address to identify the country one is visiting from, and to infer language to be used for personalisation. Note that this example uses [[DPV-PD]] for personal data concepts.

```
ex:PersonaliseWebsiteForVisitors rdf:type dpv:PersonalDataHandling ;
    dpv:hasPurpose dpv:Personalisation ;
    dpv:hasPersonalData dpv-pd:BrowserFingerprint,
                        dpv-pd:IPAddress ;
    dpv:hasProcessing dpv:Collect, dpv:Use ;
    dpv:hasPersonalDataHandling ex:DeriveVisitorCountry ;
    dpv:hasPersonalDataHandling ex:InferVisitorLanguage .

ex:DeriveVisitorCountry rdf:type dpv:PersonalDataHandling ;
    dpv:hasProcessing dpv:Derive ;
    dpv:hasPersonalData ex:VisitorCountry .
ex:VisitorCountry rdf:type dpv:DerivedPersonalData ;
    skos:broaderTransitive dpv-pd:Country .

ex:InferVisitorLanguage rdf:type dpv:PersonalDataHandling ;
    dpv:hasProcessing dpv:Derive ;
    dpv:hasPersonalData ex:VisitorCountry .
ex:VisitorLanguage rdf:type dpv:InferredPersonalData ;
    skos:broaderTransitive dpv-pd:Language .
```

Technical and Organisational Measures

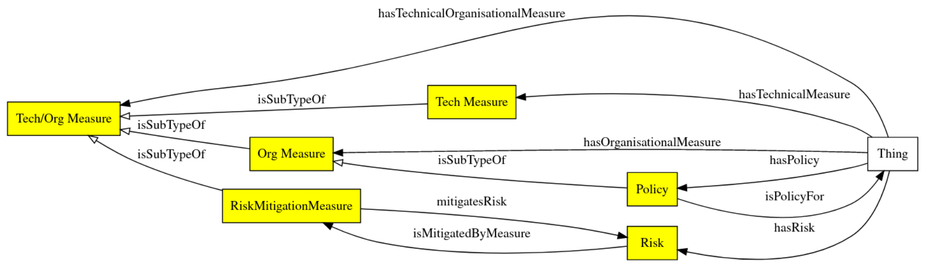

Figure 24: Overview of Technical and Organisational Measure concepts in DPV

DPV's taxonomy of tech/org measures are structured into two groups representing technical and organisational measures respectively. In addition to the broad categories `TechnicalMeasure` and `OrganisationalMeasure`, DPV also provides concepts `Policy` for representing policies in place, and `Risk` for describing applicable risks and their management or mitigation. Similar to use of other concepts, it is advised for an adopter to identify and use or extend the relevant concept(s) to represent measures relevant to their use-case.

DPV is looking to enrich its taxonomy of technical and organisational measures through adoption of existing standards, best practices, and widely relevant practices. For this, we welcome contribution of concepts from sources such as ISO/IEC standards, ENISA, NIST, IETF, and others.

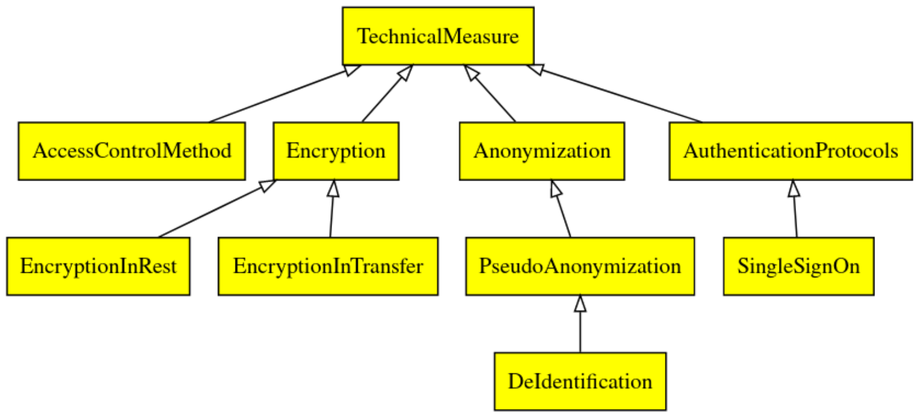

Figure 25: Overview of Technical Measures in DPV

Technical Measures To indicate data is encrypted using the Rivest-Shamir-Adleman (RSA) method, one would extend the `Encryption` concept within DPV to represent `RSA`, and then instantiate it with the specific implementation used (e.g. to indicate key size). Access to this data is further restricted by requiring a password or credential.

```
ex:RSAEncryption rdf:type dpv:TechnicalMeasure ;
    skos:broaderTransitive dpv:Encryption ;
    skos:scopeNote "Key size: 2048, Custom Implementation"@en .
ex:RBACCredential rdf:type dpv:TechnicalMeasure ;
    skos:broaderTransitive dpv:AccessControlMethod .
```

Organisational Measures To indicate staff are trained in the use of credentials, and that a policy exists regarding this, the use of `OrganisationalMeasure` concepts can be combined in several ways. Note that the interpretations for how staff training is associated with credentials, or contains training regarding

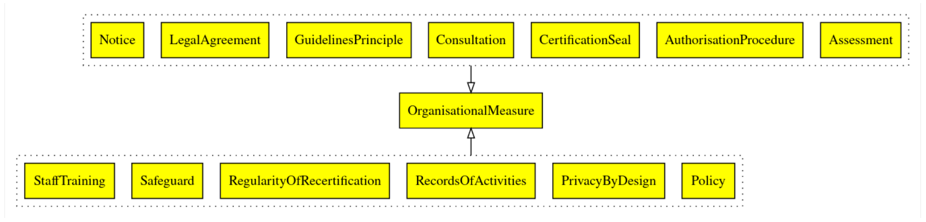

Figure 26: Overview of Technical Measures in DPV

credentials is arbitrary in notation. It is intended to demonstrate how different perspectives can be represented so as to be suitable to the organisation's documentation practices.

```
# 1: directly associating staff training with credentials used
ex:StaffCredentialsTraining rdf:type dpv:OrganistionalMeasure ;
    skos:broaderTransitive dpv:StaffTraining .
ex:RBACCredential dpv:hasOrganisationalMeasure ex:StaffCredentialTraining .

# 2: security policy containing staff training and access control
ex:SecurityPolicy rdf:type dpv:OrganisationalMeasure ;
    skos:broaderTransitive dpv:Policy ;
    dpv:hasOrganisationalMeasure dpv:StaffTraining ;
    dpv:hasTechnicalMeasure dpv:AccessControlMethod .

# 3: indicating staff training contains access control methods
ex:StaffCredentialsTraining rdf:type dpv:OrganistionalMeasure ;
    skos:broaderTransitive dpv:StaffTraining ;
    dpv:hasTechnicalMeasure ex:RBACCredential .
```

Risk and Risk Mitigation For risk management, DPV's `Risk` and `hasRisk` are useful to indicate applicability or existence of a risk, along with `RiskMitigationMeasure` and `mitigatesRisk` to indicate its mitigation. This permits a rudimentary association of risks and mitigations, through technical and organisational measures, within a use-case. In this, it is important to note that the DPV is not a risk management vocabulary as it does not model several of the concepts necessary for risk assessment and documentation. For more developed representations of risk assessment, mitigation, and management vocabularies, we suggest the adoption of relevant standards, such as the ISO/IEC 31000 series, and welcome contribution for their representation within DPV.

Risk, as a concept, can be associated with any other concept given its broad existence and applicability, whereas its mitigation is defined as a technical and organisational measure. Through this, the implemented or adopted technical and organisational measures can be annotated with the risks they address or mitigate. For example, the storage of personal data within a database where the implementation of access control methods mitigates the risks of unauthorised access is described as follows.

```
# 1: annotating implementations with risks involved
ex:DataStore rdf:type dpv:Technology ;
    dpv:hasTechnicalMeasure ex:RBACCredential ;
    dpv:hasRisk ex:UnAuthorisedAccess .

# 2: risk registry denoting risks and mitigations
ex:UnAuthorisedAccess rdf:type dpv:Risk ;
    dpv:isMitigatedByMeasure ex:RBACCredential .

# 3: annotating measures with risks mitigated
ex:RBACCredential rdf:type dpv:TechnicalMeasure, dpv:RiskMitigationMeasure ;
    skos:broaderTransitive dpv:AccessControlMethod ;
    dpv:mitigatesRisk ex:UnAuthorisedAccess .

# 4: policies and training as risk mitigations
ex:SecurityPolicy rdf:type dpv:OrganisationalMeasure ;
    skos:broaderTransitive dpv:Policy ;
    dpv:hasOrganisationalMeasure dpv:StaffTraining ;
    dpv:mitigatesRisk ex:UnAuthorisedAccess .
```

Data Processing Agreements The term *Data Processing Agreement* refers to a broad concept related to contracts or agreements between entities representing conditions regarding the processing of (personal-)data. This can include ad-hoc 'data handling' policies such as NDAs, embargoes, and enforcement of practices, as well as more formal and legal binding contractual obligations such as those between a Controller and a Processor.

To represent such concepts, DPV provides `LegalAgreement`, along with sub-types for `NDA` (Non-disclosure agreements), `ContractualTerms`, and `DataProcessingAgreement`. In these, it is important to remember that while *contract* can also be a form of legal basis, the concept represented here is not necessarily the same *contract* as that used to justify the processing of personal data with a data subject. Here, *contracts* are a broad category representing contractual terms governing data handling within or with an entity.

For representing specific agreements between entities (other than those with data subjects - which are covered in Legal Basis taxonomy), DPV provides the following types of agreements:

- `ControllerProcessorAgreement`: An agreement between a Controller and a Processor, where the Controller instructs the Processor(s) to carry out processing on its behalf.

- `JointControllersAgreement`: An agreement between two or more Controllers to act as a 'Joint Controller'.
- `SubProcessorAgreement`: An agreement between two or more Processors where one Processor instructs another to carry out processing on its behalf.
- `ThirdPartyAgreement`: An agreement between a Data Controller or a Data Processor, and a Third Party. Note that this is a loosely defined concept, as depending on the jurisdiction, this relationship may result in the Third Party being a Data Controller or a Joint Data Controller.

To indicate the entities involved in an agreement, the relation `hasEntity` can be used, or relations associated with specific roles to indicate contextuality. For example, using `hasDataController` with a `ControllerProcessorAgreement` denotes the Data Controller for that agreement.

Acme is the Data Controller, that contracts BetaInc as a Data Processor to analyse raw call logs and provide statistical patterns.

```
ex:Acme rdf:type dpv:DataController .
ex:BetaInc rdf:type dpv:DataProcessor .

ex:AcmeBetaContract rdf:type dpv:ControllerProcessorAgreement ;
    dpv:hasDataController ex:Acme ;
    dpv:hasDataProcessor ex:Beta ;
    # part1: acme sends data to beta
    dpv:hasPersonalDataHandling ex:AcmeProvision ;
    # part2: beta sends data to acme
    dpv:hasPersonalDataHandling ex:BetaProvision .

ex:AcmeProvision rdf:type dpv:PersonalDataHandling ;
    skos:note "Acme transfers call logs to Beta"@en ;
    dpv:hasPersonalData ex:CallLogs ;
    dpv:hasProcessing ex:TransferCallLogs ;
    dpv:hasDataController ex:Acme ;
    dpv:hasDataProcessor ex:BetaInc .

ex:BetaProvision rdf:type dpv:PersonalDataHandling ;
    skos:note "Beta analyses and transfers call statistics to Acme"@en ;
    dpv:hasPersonalData ex:CallStatistics ;
    dpv:hasProcessing ex:AnalyseCalls, dpv:TransferStatistics ;
    dpv:hasDataProcessor ex:BetaInc ;
    dpv:hasRecipientDataController ex:Acme ;
    # alternative way to explicitly indicate who is implementing this
    dpv:isImplementedByEntity ex:BetaInc .

ex:CallLogs rdf:type dpv:PersonalData ;
    skos:broaderTransitive dpv-pd:CallLog ;
    # denoting source of data as part of agreement
    dpv:hasDataSource ex:Acme .
ex:CallStatistics rdf:type dpv:DerivedData ;
    skos:broader ex:CallLogs ;
    dpv:hasDataSource ex:BetaInc .

ex:TransferCallLogs rdf:type dpv:Processing ;
    skos:broaderTransitive dpv:Transfer ;
    dpv:hasPersonalData ex:CallLogs ;
    # alternative 1 - based on implementation and recipient
    dpv:isImplementedByEntity ex:Acme ;
    dpv:hasRecipient ex:BetaInc ;
    # alternative 2 - based on data exporter/importer roles
    dpv:hasDataExporter ex:Acme ;
    dpv:hasDataImporter ex:BetaInc .

ex:AnalyseCalls rdf:type dpv:Processing ;
    skos:broaderTransitive dpv:Analyse ;
    # no recipients, data is analysed by BetaInc
    dpv:isImplementedByEntity ex:BetaInc .

ex:TransferStatistics rdf:type dpv:Processing ;
    skos:broaderTransitive dpv:Transfer ;
    dpv:hasPersonalData ex:CallStatistics ;
    # using alternative 2 from above
    dpv:hasDataExporter ex:BetaInc ;
    dpv:hasDataImporter ex:Acme .
```

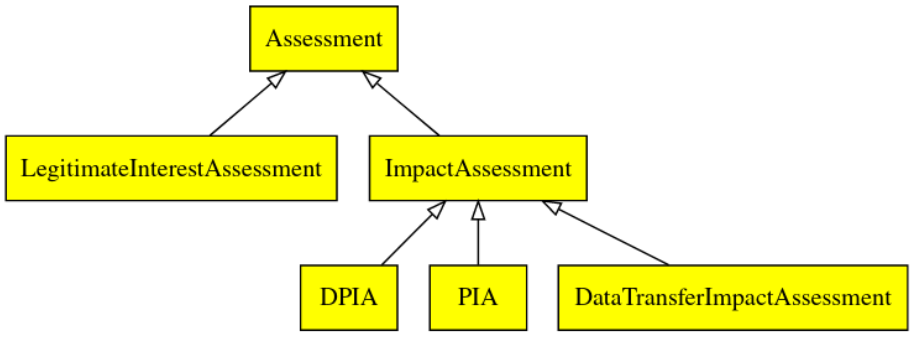

Figure 27: Types of Impact Assessments in DPV

Impact Assessments

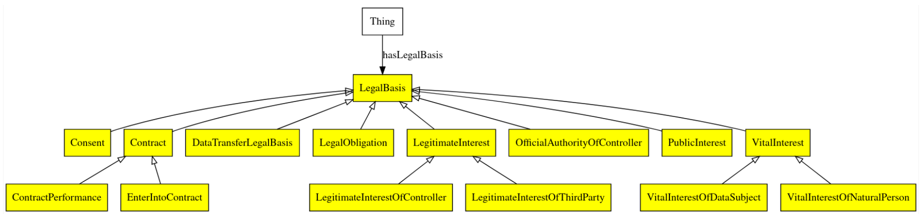

Figure 28: Overview of Legal Basis concepts in DPV

Legal Basis

DPV provides the following categories of legal bases based on [[GDPR]] Article 6: consent of the data subject, contract, compliance with legal obligation, protecting vital interests of individuals, legitimate interests, public interest, and official authorities. Though derived from GDPR, these concepts can be applied for other jurisdictions and general use-cases. The legal bases are represented by the concept `LegalBasis` and associated using the relation `hasLegalBasis`.

When declaring a legal basis, it is important to denote under what law or jurisdiction that legal basis applies. For instance, using `Consent` as a legal basis has different obligations and requirements in EU (i.e. [[GDPR]]) as compared to other jurisdictions. Therefore, unless the information is to be implicitly interpreted through some specific legal lens or jurisdictional law, DPV recommends indicating the specific law or legal clause associated with the legal basis so as to scope its interpretation. This can be done using the relation `hasJurisdiction` or `hasLaw`.

For GDPR, DPVCG provides the [[[DPV-GDPR]]] which defines the legal bases within [[GDPR]] by extending them from relevant concepts within the DPV. We welcome similar contributions for extending the GDPR extension as well as creating extensions for other laws and domains.

The `LegalBasis` can be associated with any concept using the relation `hasLegalBasis`. Such associations are of three types: (1) where the legal basis refers to an instance, such as the consent or contract associated with a particular data subject; (2) where the legal basis refers to the category that will be used to justify processing, such as the concept *consent* to denote consent will be the basis for indicated processing; and lastly (3) where the legal basis is the denoted with context, such as *consent of service consumers*.

```
# 1: instance of legal basis
# interpretation: consent of user #00145667 is the legal basis
ex:PDH1 rdf:type dpv:PersonalDataHandling ;
    dpv:hasDataController ex:Acme ;
    dpv:hasDataSubject ex:ServiceConsumers ;
    dpv:hasLegalBasis ex:ConsentUserN00145667 .
ex:ConsentUserN00145667 rdf:type dpv:Consent ;
```

```
    dpv:hasDataSubject ex:UserN00145667 .

# 2: category of legal basis
# interpretation: consent is the legal basis
ex:PDH1 rdf:type dpv:PersonalDataHandling ;
    dpv:hasDataController ex:Acme ;
    dpv:hasDataSubject ex:ServiceConsumers ;
    dpv:hasLegalBasis dpv:Consent .

# 3: category of legal basis with contextual information
# interpretation: consent of service consumers is the legal basis
ex:PDH1 rdf:type dpv:PersonalDataHandling ;
    dpv:hasDataController ex:Acme ;
    dpv:hasDataSubject ex:ServiceConsumers ;
    dpv:hasLegalBasis ex:UserConsent .
ex:UserConsent rdf:type dpv:LegalBasis ;
    skos:broaderTransitive dpv:Consent ;
    dpv:hasDataSubject ex:ServiceConsumers .
```

When using legal bases, we advice careful consideration whether the information to be represented is regarding a specific instance (e.g. consent of an individual) or a general category (e.g. contract of service consumer/users), and to utilise DPV concepts accordingly.

Consent

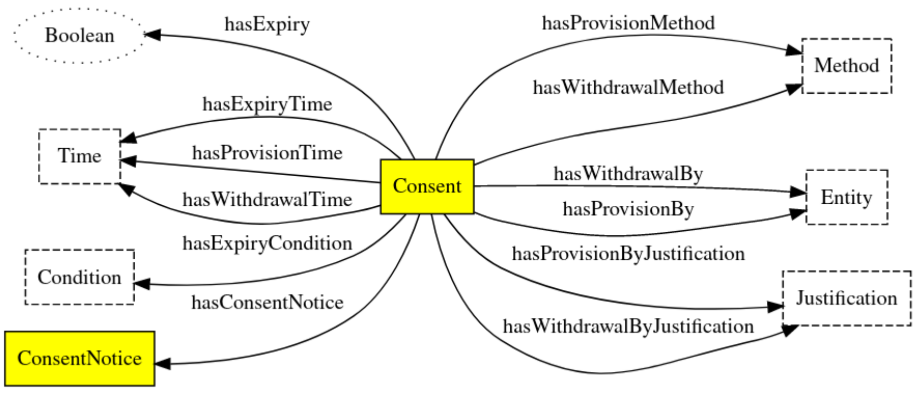

Figure 29: Denoting information using `Consent`

`Consent` is a specific (and perhaps special) legal basis given its emphasis on individual empowerment and control, as well as the attention and relevance it receives from being in front of the individuals via ubiquitous consent dialogues throughout the web. DPV provides concepts for representing information about how consent, as a legal basis, is utilised (by the Controller), provided or given (by the Data Subject), how long it is considered to be valid (its duration), and how it can be withdrawn. This information can be utilised in applications associated with consent, such as creating a ‘record’ of consent for compliance.

Given the reliance of consent as a legal bases whose validity is associated with requirements and obligations based on jurisdictional laws, DPV does not dictate what constitutes a ‘valid’ consent and only provides a way to declare information about it. Such information concerning compliance obligations and requirements is within the scope of the DPVCG, and we welcome contributions on how this can be represented in a coherent manner within an extension that is compatible with the rest of DPV.

The concept `Consent` can be used as is or with another concept as its legal basis e.g. `PersonalDataHandling` and `hasLegalBasis`. Similarly, the relevant information, such as purposes or personal data for which consent is expressed can be associated with consent or the concept it is used within using relations such as `hasPurpose` and `Purpose`.

In the first example, consent is directly associated with purposes and personal data handling and entities. This is relevant where information about consent needs to be documented in the form of ’consent records’ or an equivalent use-case.

```
ex:UserConsent rdf:type dpv:LegalBasis ;
    skos:broaderTransitive dpv:Consent ;
    dpv:hasDataController ex:Acme ;
    dpv:hasDataSubject ex:ServiceConsumers ;
    dpv:hasPersonalData dpv:Email ;
    dpv:hasLegalBasis ex:UserConsent .
```

In the second example, consent is utilised within a `PersonalDataHandling` instances as its legal basis, and is associated with purposes and personal data handling and entities through it. This is relevant where information about consent is documented amongst other legal bases.

```
ex:PDH rdf:type dpv:PersonalDataHandling ;
    dpv:hasDataController ex:Acme ;
    dpv:hasDataSubject ex:ServiceConsumers ;
    dpv:hasPersonalData dpv:Email ;
    dpv:hasLegalBasis ex:UserConsent .
```

DPVCG intends to update/upgrade the consent section of DPV to enable a broader representation of information associated with consent and relevant information associated with it. We welcome contributions and discussions for the same.

Consent Notice, Provision, and Withdrawal Requirements for *informed* consent require provision of information before the consent is obtained so as to inform the individual. This information is typically provided through a *notice*, which can be specified using the property `hasConsentNotice`.

To specify additional information, DPV specifies provenance properties of `hasProvisionTime` and `hasProvisionMethod` for how consent is given, and `hasWithdrawalTime` and `hasWithdrawalMethod` to indicate how consent was withdrawn. The expiry of consent can be specified using the property `hasExpiry`, with further specific types related to expiry as a temporal entity (e.g. duration) using `hasExpiryTime` or as a condition/event using `hasExpiryCondition`.

```
ex:UserConsent rdf:type dpv:Consent ;
    # consent provision
    dpv:hasProvisionMethod "form signature"@en ;
    dpv:hasProvisionBy dpv:DataSubject ;
    dpv:hasProvisionTime "1969-01-01T00:00:00"^^xsd:dateTime ;

    # consent expiry condition
    dpv:hasExpiryTime "1972-12-31T23:59:59"^^xsd:dateTime ;
    dpv:hasExpiryCondition "account closure"@en ;

    # consent withdrawal
    dpv:hasWithdrawalMethod "telephone call"@en ;
    dpv:hasWithdrawalBy dpv:DataSubject ;
    dpv:hasWithdrawalTime "1969-12-31T00:00:00"^^xsd:dateTime ;

    dpv:hasConsentNotice "printed form provided before study"@en .
```

Explicit Consent By default, consent is expected to adhere to several requirements such as being informed, freely given, and so on - typically defined within law or other relevant guidelines, that determine its validity. *Explicit* consent is a specific form of consent where the action of giving consent carries additional obligations of being explicitly performed by the individual (data subject). To represent this, the relation `isExplicit` is provided with the suggestion to use it as a boolean indicator or flag i.e. *True* indicates consent is explicit.

The specific conditions under which consent can be considered valid, or informed, or explicit as defined under jurisdictional law. The terms provided within the DPV are generic concepts for representing these *types* of consent. For using specific jurisdictional requirements and use of consent as a legal basis, it is advisable to declare such jurisdictional concepts in a separate extension to the DPV. For specific use of consent as a legal basis defined within GDPR, we provide [[DPV-GDPR]].

```
ex:UserConsent rdf:type dpv:Consent ;
```

```
    dpv:isExplicit true .  # xsd:boolean
```

Consent by Delegation To specify consent provided by delegation, such as in the case of a parent or guardian providing consent for a child, the properties `hasProvisionByJustification` and `hasWithdrawalByJustification` can be used to capture the nature of such 'delegation', with the fields `hasProvisionBy` and `hasWithdrawalBy` representing the legal entity who provided the consent. By default, the consent can be assumed to be provided by the associated Data Subject.

Legal Bases for Data Transfers Given the importance of data transfers or movement across jurisdictional borders, DPV also defines the concept `DataTransferLegalBasis` to represent the category of legal bases applicable for justifying such cross-border data flows. For GDPR, these involve Articles 45, 46, and 49, and are defined within the [[DPV-GDPR]] extension. When representing such cross-border data flows, concepts such as jurisdiction and countries are also required so as to indicate which laws apply, and whether political treaties exist between regions. We welcome contributions for expressing this information within the framework of the DPV.

In this example, Acme is a Data Controller in Ireland, that engages BetaInc, a Data Processor based in Canada - a country with adequacy decision (for GDPR), for transferring credit card numbers to perform payment transactions. While the payment has its own legal basis (contract), the data transfer is further justified with a separate legal basis (adequacy decision from [[DPV-GDPR]]) with explicit notation of the data importer/exporter roles.

In these, `dpv-juris:EUAdequacyCA` represents the adequacy decision based on `dpv-gdpr:A45-3` between EU and Canada, and is part of the proposed *juris* extension modelling countries and data transfer agreements between them.

```
ex:Acme rdf:type dpv:LegalEntity ;
    dpv:hasLocation dpv-juris:Ireland .
ex:BetaInc rdf:type dpv:LegalEntity ;
    dpv:hasLocation dpv-juris:Canada .

ex:PDH rdf:type dpv:PersonalDataHandling ;
    dpv:hasPurpose dpv:Payment ;
    dpv:hasPersonalData dpv-pd:CreditCardNumber ;
    dpv:hasProcessing ex:PaymentDataTransfer ;
    dpv:hasDataController ex:Acme ;
    dpv:hasDataProcessor ex:BetaInc ;
    dpv:hasLegalBasis dpv:Contract .

ex:PaymentDataTransfer rdf:type dpv:Processing ;
    skos:broaderTransitive dpv:Transfer ;
    dpv:hasDataExporter ex:Acme ;
    dpv:hasDataImporter ex:BetaInc ;
    dpv:hasLegalBasis dpv-juris:EUAdequacyCA .
```

Entities

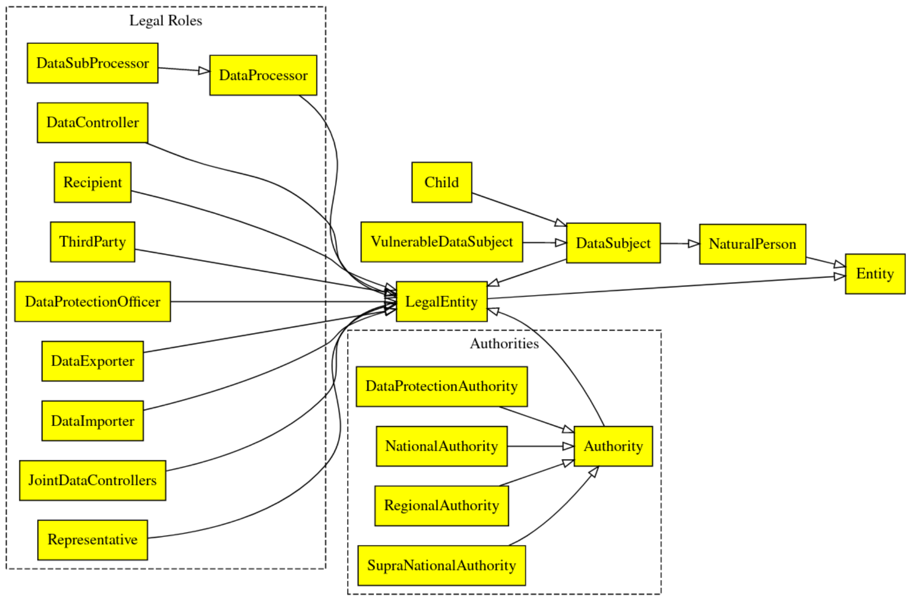

Figure 30: Overview of Entities in DPV

A 'legal entity' is an entity whose role is defined legally or within legal norms. DPV provides a taxonomy of entities based on their application within laws and use-cases. Alongside `DataController` and `DataSubject`, the DPV also defines other types of entities involved in the processing of personal data such as `DataProcessor`, `ThirdParty`, or terms associated with data transfers such as `DataImporter` and `DataExporter`. In addition to these, various kinds of `Authority`, including `DataProtectionAuthority` are also provided.

Entity Descriptions

```
ex:Acme rdf:type dpv:LegalEntity ;
    dpv:hasName "Acme"@en ;
    dpv:hasAddress "Dublin, Ireland"@en ;
    dpv:hasContact "acme@example.com" ;
    dpv:hasRepresentative ex:AcmeDPO,  # internal DPO
                          ex:AcmeEUOrg ; # EU Representative
    dpv:hasLocation iso:IE, # if an ISO vocabulary for country-codes is used
```

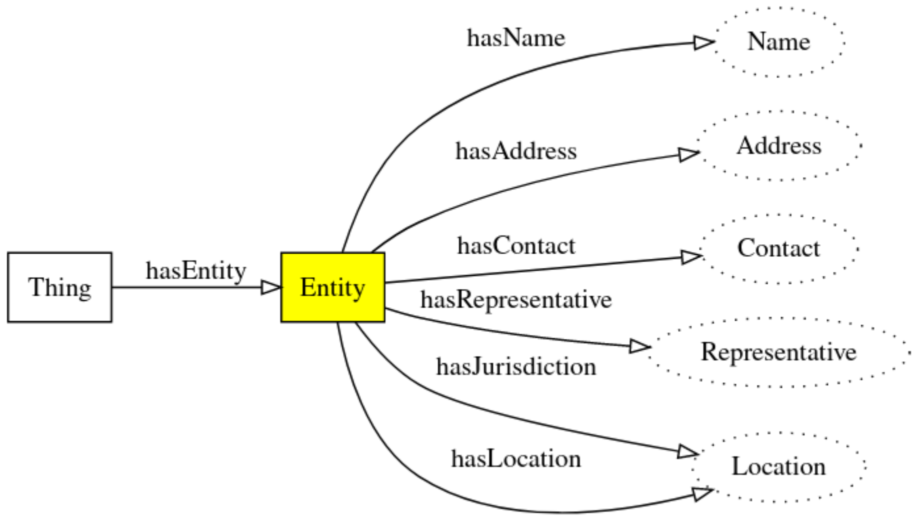

Figure 31: Describing Entities in DPV

```
                    dpv-juris:Ireland, # dpv-juris proposed vocabulary
                    "Ireland"@en . # plain strings

ex:AcmeDPO rdf:type dpv:Representative ;
    dpv:hasEntity ex:Acme .
ex:AcmeEUOrg rdf:type dpv:LegalEntity, dpv:DataProtectionOfficer ;
    dpv:hasEntity ex:Acme ;
    dpv:hasLocation "EU"@en .
```

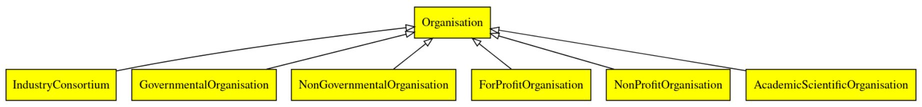

Figure 32: Describing Entities in DPV

Organisation Categories

Contextual Information

For indicating additional information regarding how the expressed information should be interpreted, or how it applies within a particular *context*, the `Context` concept along with the `hasContext` relationship can be used. `Context` refers to a generic collection of concepts that assist in indicating information such as the necessity, importance, environment - which aid in the interpretation or application of other core concepts.

Importance and Necessity Currently DPV provides two subtypes of concepts - `ContextualImportance` and `ContextualNecessity`, which can be applied to `PersonalDataHandling`, `Purpose`, `PersonalData`, or other relevance concepts. `ContextualNecessity` requires to whether something is *Required*, is *Optional, or is `NotRequired`. These can be used to indicate, amongst other things, whether personal data is optional, or if a particular processing is required to be carried out.*

In this example, a `PersonalDataHandling` instance is comprised of two more `PersonalDataHandling` instances for each of the optional and required parts. This means that for the stated purpose, an Account Identifiers is required, but the use of Email is optional.

Example using [[DPV-SKOS]]

```
:PDH1 a dpv:PersonalDataHandling ;
  # optionally also declare data and purpose for PDH1
  dpv:hasPersonalData dpv:Email, dpv:AccountIdentifier ;
  dpv:hasPurpose dpv:CommunicationForCustomerCare ;
  dpv:hasPersonalDataHandling :PDH2, :PDH3 .

:PDH2 a dpv:PersonalDataHandling ;
  dpv:hasContext dpv:Optional ;
  dpv:hasPersonalData :Email ;
  dpv:hasPurpose dpv:CommunicationForCustomerCare .

:PDH3 a dpv:PersonalDataHandling ;
  dpv:hasContext dpv:Required ;
  dpv:hasPersonalData :ID ;
  dpv:hasPurpose dpv:CommunicationForCustomerCare .
```

Identifiers, Location, and Duration To indicate an identifier for a concept, such as its registration number or internal reference, or other forms of defined names, the relation `hasIdentifier` can be used. This permits entities, processing operations, purposes, personal data handling instances, and other concepts to be represented along with their identifiers used within a use-case or organisation.

To specify the location and duration of concepts, the relations `hasLocation` and `hasDuration` can be used along with `Location` and `Duration` concepts. The processing taxonomy extends these to specify `StorageLocation` and `StorageDuration` as specific concepts relevant to the storage of personal data. Similarly, other concepts can utilise these either as-is or by extending/specialising them, such as to indicate the duration of purposes, or the location of data transfers.

Technology, Consequence To indicate how something is implemented, or who is implementing it, the relations `isImplementedUsingTechnology` and `isImplementedByEntity` can be used. These are associated with `Technology` and `Entity` respectively.

To denote the consequences of something, such as a processing operation or a purpose, the concept `Consequence` and the relation `hasConsequence` can be used

.

For further contextual information, additional concepts and relationships provided include: `AlgorithmicLogic` and `hasAlgorithmicLogic`; and `HumanInvolvement` and `hasHumanInvolvement`. These are important considerations in impact assessments and automated decision making.

Contributing to DPV

The DPVCG welcomes participation regarding the DPV, including expansion or refinement of its terms, addressing open issues, and welcomes suggestions on their resolution or mitigation.

While we welcome participation via any and all mediums - e.g., via GitHub pull requests or issues, emails, papers, or reports - the formal resolution of contributions takes place only through the DPVCG meeting calls and mailing lists. We therefore suggest joining the group to participate in these discussions for formal approval.

For contributions to the DPV, please see the section on GitHub. The current list of open issues and their discussions to date can be found at GitHub issues. Note, GitHub Issues are preferred for discussion of concepts and proposals.

To suggest a new term, we request following information:

1. term
2. description of the term
3. whether it should be a class or a property
4. relation to existing term(s) in DPV e.g. through sub-classes
5. source (where applicable)
6. justification or relevance of why this term should be added (where not obvious)

Notes

This document is based on inspiration from the following:

- RDF 1.1 Primer https://www.w3.org/TR/rdf11-primer/

- OWL 2 Primer https://www.w3.org/TR/owl2-primer/
- PROV Model Primer https://www.w3.org/TR/prov-primer/

Acknowledgements

DPVCG was initiated as part of the SPECIAL H2020 Project, which received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No. 731601. This work and Harshvardhan J. Pandit have been funded by the Irish Research Council Government of Ireland Postdoctoral Fellowship Grant #GOIPD/2020/790.

Thanks to Georg P Krog (Signatu), Piero Bonatti (Universita' di Napoli Federico II), and Beatriz Esteves (Universidad Politécnica de Madrid) for their comments on drafts of this document.