Chapter 1: ODRL

1.1 Il linguaggio

1.1.1 Definizione ed obiettivi

"L' Open Digital Rights Language (ODRL) è un linguaggio per l'espressione di policy che definisce: un modello dell'informazione flessibile ed interoperativo, un vocabolario e un meccanismo di codifica per la rappresentazione delle istruzioni sull'uso di contenuti o servizi"[4].
Il linguaggio si pone all'interno dello scenario applicativo nel quale vi è la necessità di definire:

- quali azioni siano permesse o proibite su una risorsa. Queste regole possono essere imposte da leggi o direttamente dal possessore dell'asset o servizio;
- indicare quali attori interagiscono con le policy definite; in particolare chi può definire le policy e a chi si applicano;
- indicare eventuali limiti riguardanti i permessi ed i divieti espressi;.

Avere un modello standard per definire questi bisogni dà 2 fondamentali vantaggi:

- chi possiede l'asset è in grado di definire in modo chiaro quali siano le azioni che un consumatore può fare, evitanto quindi usi indesiderati;
- chi usufruisce dell'asset conosce in modo preciso quali azioni può compiere, evitando così di infrangere regole o leggi.

ODRL definisce un modello semantico di permessi, divieti ed obblighi, che può essere usato per descrivere le modalità d'uso di un contenuto. In particolare si cerca di definire i concetti chiave per la creazione di policy machine-readable collegate direttamente all'asset al quale sono associate, permettendo all'utente finale di reperire facilmente informazioni sulla risorsa che utilizza. Quest'ultimo requisito è soddisfatto, poiché ODRL è costruito seguendo i *Linked Data principles*[1]:

- Utilizzo di URIs come nomi per le risorse;
- Gli URI sono indirizzi HTTP in modo che le persone possano cercare informazioni sulle risorse;
- L'URI deve fornire informazioni utili sulla risorsa;
- Tra le informazioni, fornire altri URI, in modo che l'utente possa raggiungere altre informazioni.

Nonostante questi principi siano più indicati per un'implementazione graph-based, è possibile anche definire utilizzi che non tengano conto dei link tra le varie informazioni.

1.1.2 Il modello

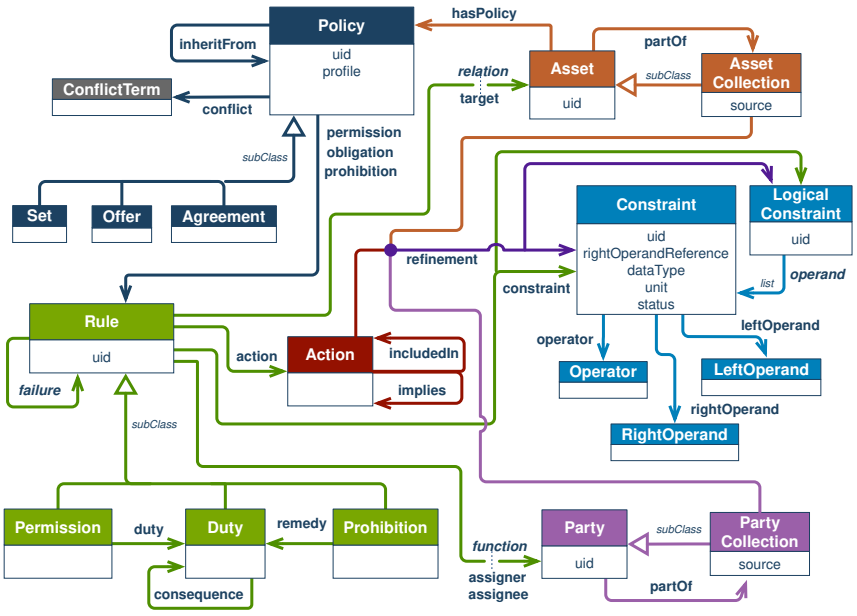

Figure 1.1: Schema del modello ODRL[4]

Come visibile all'interno dello schema in figura 1.1, il modello è basato sulle seguenti entità principali:

- **Policy**: un gruppo di una o più regole;
- **Regola**: concetto astratto che racchiude le caratteristiche comuni di **permesso**, **divieto**, **doveri**;
- **Asset**: risorsa o collezione di risorse soggette a regole;

- **Azione**: operazione su un asset;
- **Party**: entità o insieme di entità con un certo ruolo in una regola;
- **Limiti**: espressione logica o booleana imposta su azioni, party, asset o regole.

Vocabolari

"L' *ODRL Vocabulary and Expression* descrive i termini usati dalle policy ODRL e come codificarle"[3]. All'interno di ODRL, i vocabolari utilizzati per definire i termini all'interno delle policy vengono detti **profili**, i quali possono essere usati per definire termini che supportano specifiche applicazioni; all'interno di un profilo è possibile, ad esempio, fornire le specifiche riguardanti nuove sottoclassi di termini già presenti nei vocabolari standard di ODRL. I 2 vocabolari principali definiti sono:

- **ODRL Core Vocabulary**: rappresenta la minima espressione di policy supportata;
- **ODRL Common Vocabulary**: arricchisce il vocabolario precedente con un gruppo di azioni generiche, nuove sottoclassi per le policy, ruoli per i party e relazioni tra gli asset.

Una delle principali differenze tra i due vocabolari, la si ha all'interno delle **azioni** che possono essere indicate: nel primo caso si hanno a disposizione solamente 2 azioni **use** e **transfer**, nel secondo caso queste 2 azioni vengono estese da diverse azioni figlie, come mostrato in figura 1.2

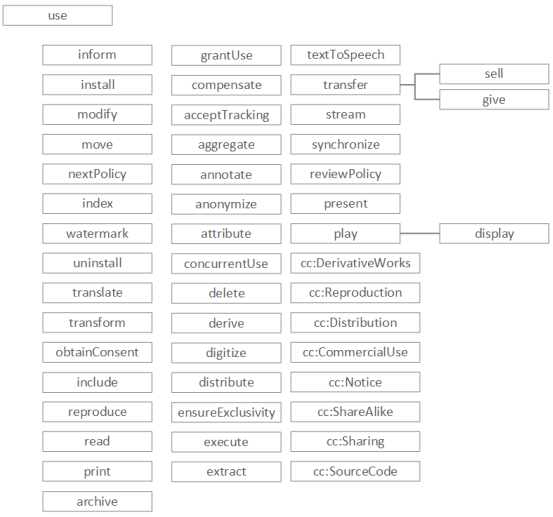

Figure 1.2: Tutte le azioni mostrate sono figlie di use, ad eccezioni di trasfer e le sue sottoazioni[5]

Policy

Come definito nel modello presente nella sezione 1.1.2, una policy è un gruppo non vuoto di **regole** e, quindi, di **permessi**, **divieti** o **obblighi**. Una policy deve soddisfare i seguenti requisiti:

- deve avere un identificativo univoco, detto **uid**;
- deve avere almeno una regola;
- può specificare un profilo, obbligatorio se non si usa il Core Vocabulary mostrato nella sezione 1.1.2;
- può specificare una policy da cui eredita le proprietà;
- può specificare una strategia per la risoluzione dei conflitti.

Come visibile dalla figura 1.1, una policy ha 3 prossibili sottoclassi:

- **Set**: un insieme di regole che hanno effetto;
- **Agreement**: regole concesse ad una entità assegnataria da una assegnatrice;
- **Offer**: proposta di una regola da parte di un assegnatore.

Di seguito un esempio di policy definita mediante ODRL:

```
{
 "@context": "http://www.w3.org/ns/odrl.jsonld",
 "@type": "Set",
 "uid": "http://example.com/policy:1010",
 "permission": [{
   "target": "http://example.com/asset:9898.movie",
   "action": "use"
 }]
}
```

Listing 1.1: Policy con sottoclasse **Set**

La policy mostrata nel listing 1.1 presenta i campi:

- @type: serve ad indicarne la sottoclasse;
- @context : serve ad indicare che il file deve essere conforme ad ODRL, rappresentato dall'URL da cui si può ottenere l'ODRL Common Vocabulary[2];

- nel contesto sono presenti altri link per le altre proprietà, ad esempio quello per la descrizione di *Set* e per *use*;
- non usando termini fuori dai 2 vocabolari principali, non necessita la definizione di un profilo;
- l'id univoco è rappresentato da un URL che porta alle informazioni relative la risorsa.

Asset

Come definito nel modello presente nella sezione 1.1.2, un asset è una risorsa o una collezione di risorse soggette a regole. Un asset può essere una qualunque risorsa identificabile. Ha **AssetCollection** come sottoclasse, la quale rappresenta una collezione di asset. La classe asset può avere:

- un identificativo univoco, il quale può essere omesso se l'asset è fornito direttamente con la policy; le specifiche di ODRL, pur supportando questo caso d'uso, sconsigliano questa pratica;
- una o più proprietà denominate **partOf**: identifica le collezioni di cui fa parte l'asset, il quale può essere a sua volta una collezione.

Esempio di utilizzo in una regola:

```
{
"@context": "http://www.w3.org/ns/odrl.jsonld",
 "@type": "Offer",
 "uid": "http://example.com/policy:3333",
 "profile": "http://example.com/odrl:profile:02",
 "permission": [{
        "target": "http://example.com/asset:3333",
        "action": "display",
        "assigner": "http://example.com/party:0001"
 }]
}
```

Listing 1.2: Utilizzo di asset nella proprietà **target** di una regola

La sottoclasse **AssetCollection** può avere i seguenti 2 campi aggiuntivi:

- **source**: sostituisce il campo **uid** nelle classe AssetCollection all'interno di un **refinement**;

- una o più **refinement**: limiti riguardanti la collezione che identificano solamente un sottogruppo di asset al suo interno.

Esempio di utilizzo della proprietà **partOf**:

```
{
        "@type": "dc:Document",
        "@id": "http://example.com/asset:111.doc",
        "dc:title": "Annual Report",
        ...
        "odrl:partOf": "http://example.com/archive1011",
        ...
}
```

Listing 1.3: L'asset definito è parte del target presente nel listing 1.2

In quest'ultimo esempio si ha che l'asset con id "`http://example.com/asset:111.doc`" è definito come parte della collezione "`http://example.com/archive1011`". Questa definizione ha come effetto l'applicarsi della policy presente nel listing 1.2 anche all'asset che fa parte della collezione.

Party

Come definito nel modello presente nella sezione 1.1.2, un party è una entità o una collezione di entità con una funzione determinata in una regola. Un party può essere un qualunque soggetto con un ruolo attivo nelle regole o che produce un effetto specifico, ad esempio controlla che le azione relative ad un dovere vengano effettuate. Ha **PartyCollection** come sottoclasse, la quale rappresenta una collezione di entità. La classe party può avere:

- un identificativo univoco, il quale può essere omesso se è possibile definire in altro modo l'entità; le specifiche di ODRL, pur supportando questo caso d'uso, sconsigliano questa pratica;
- una o più proprietà denominate **partOf**: identifica le collezioni di cui fa parte l'entità, la quale può essere a sua volta una collezione.

Esempio di utilizzo in una regola:

```
{
"@context": "http://www.w3.org/ns/odrl.jsonld",
"@type": "Agreement",
"uid": "http://example.com/policy:8888",
```

```
"profile": "http://example.com/odrl:profile:04",
"permission": [{
"target": "http://example.com/music/1999.mp3",
"assigner": "http://example.com/org/sony-music",
"assignee": "http://example.com/people/billie",
"action": "play"
}]
}
```

Listing 1.4: Utilizzo di party nelle proprietà **assigner** ed **assignee** di una regola

La sottoclasse **PartyCollection** può avere i seguenti 2 campi aggiuntivi:

- **source**: sostituisce il campo **uid** nelle classe PartyCollection all'interno di un **refinement**;
- una o più **refinement**: limiti riguardanti la collezione che identificano solamente un sottogruppo di asset al suo interno.

Esempio di utilizzo della proprietà **partOf**:

```
{
"@type": "vcard:Individual",
"@id": "http://example.com/person/murphy",
"vcard:fn": "Murphy",
"vcard:hasEmail": "murphy@example.com",
...
"odrl:partOf": "http://example.com/team/A",
...
}
```

Listing 1.5: L'entità definita è parte di una PartyCollection

In quest'ultimo esempio si ha che l'entità con id "`http://example.com/person/murphy`" è definita come parte della collezione "`http://example.com/team/A`". Questa definizione ha come effetto l'affidare le funzioni di quest'ultima collezione anche alla singola entità che ne fa parte.

Action

Come definito nel modello presente nella sezione 1.1.2, **action** è una classe che rappresenta un'operazione che può essere esercitata su un asset, al quale viene associata mediante la propietà **action** di una regola. Nell' ODRL Core Vocabulary sono presenti 2 azioni principali:

- use: un qualsiasi utilizzo dell'asset;
- transfer: una qualsiasi azione che preveda il trasferimento di proprietà dell'asset;

Un'azione può avere le seguenti proprietà:

- refinement: raffinamenti semantici sull'azione, come ad esempio l'ammontare di un pagamento, il luogo nel quale l'azione può essere eseguita o il tempo massimo di esecuzione;
- includedIn: esprime l'azione padre; la conseguenza di questa dichiarazione risulta essere che tutte le regole applicate all'azione padre, devono valere anche per l'azione figlia;
- implies: esprime un'azione che non deve essere vietata per permettere l'azione con questa proprietà, ma le 2 azioni non hanno una relazione espressa tramite *includedIn*[1].

Come anticipato nel paragrafo 1.1.2 relativo ai profili, l'ORDL Common Vocabulary utilizza la proprietà *includedIn* per aggiungere azioni figlie sia ad *use* che *transfer*, come già mostrato nella figura 1.2.
Di seguito un esempio di azione in una regola:

```
1
2  {
3          "@context": "http://www.w3.org/ns/odrl.jsonld",
4          "@type": "Offer",
5          "uid": "http://example.com/policy:1012",
6          "profile": "http://example.com/odrl:profile:06",
7          "permission": [{
8                  "target": "http://example.com/music:1012"
                     ,
9                  "assigner": "http://example.com/org:abc",
10                 "action": "play"
11         }]
12 }
```

Listing 1.6: L'azione **play** è presente nella proprietà **action** della regola

[1] attualmente né l'ODRL Core Vocabulary né l'ODRL Common Vocabulary presentano azioni con questa proprietà

Constraint e Logical Constraint

Come definito nel modello presente nella sezione 1.1.2, **constraint** è una classe usata per comparare 2 espressioni che non sono constraint a loro volta, utilizzando un operatore relazionale. Rappresentano una limitazione tramite un confronto, la quale può essere soddisfatta o non soddisfatta. La classe presenta le seguenti proprietà:

- uno o nessun identificativo univoco, qualora si volesse riutilizzare l'espressione di confronto definita;
- un **leftOperand**: elemento a sinistra dell'operatore di confronto;
- un sottotitpo **operator**: operatore di confronto;
- uno tra:
  - **rightOperand**: elemento a destra dell'operatore di confronto, identificato direttamente;
  - **rightOperand**: elemento a destra dell'operatore di confronto, identificato con un riferimento;
- uno o nessun **dataType**: definisce il tipo dell'operando di destra;
- una o nessuna **unit**: unità di misura dell'operando di destra;
- uno o nessun **status**: per l'elemento di sinistra.

Oltre ai normali constraint, il modello definisce anche dei logical constraint, ovvero operazioni logiche su altri constraint già definiti. In questo caso le proprietà sono:

- uno o nessun identificativo univoco, qualora si volesse riutilizzare l'espressione di confronto definita;
- un sottotipo di **operand**: operatore logico tra i constraint espressi come lista al suo interno.

Esempi di utilizzi dei constraint:

```
{
 "@context": "http://www.w3.org/ns/odrl.jsonld",
 "@type": "Offer",
 "uid": "http://example.com/policy:6161",
 "profile": "http://example.com/odrl:profile:10",
 "permission": [{
  "target": "http://example.com/document:1234",
```

```
    "assigner": "http://example.com/org:616",
    "action": [{
          "rdf:value": { "@id": "odrl:print" },
          "refinement": [{
                  "leftOperand": "resolution",
                  "operator": "lteq",
                  "rightOperand": { "@value": "1200", "
                     @type": "xsd:integer" },
                  "unit": "http://dbpedia.org/resource/
                     Dots_per_inch"
          }]
          }]
   }]
}
```

Listing 1.7: Constraint su azione: l'azione **print** è permessa solo per risoluzioni minori di 1200 dpi

```
{
 "@context": "http://www.w3.org/ns/odrl.jsonld",
 "@type": "Offer",
 "uid": "http://example.com/policy:88",
 "profile": "http://example.com/odrl:profile:10",
 "permission": [{
  "target": "http://example.com/book/1999",
  "assigner": "http://example.com/org/paisley-park",
  "action": [{
    "rdf:value": { "@id": "odrl:reproduce" },
          "refinement": {
            "xone": {
              "@list": [
                    { "@id": "http://example.com/p:88/C1" },
                    { "@id": "http://example.com/p:88/C2" }
                  ]}
          }
   }]
 }]
}
```

Listing 1.8: Constraint logico su azione: l'azione **reproduce** è permessa solo nella forma di uno dei due constraint listati

```
{
"@context": "http://www.w3.org/ns/odrl.jsonld",
"@type": "Offer",
"uid": "http://example.com/policy:4444",
"profile": "http://example.com/odrl:profile:11",
"permission": [{
 "assigner": "http://example.com/org88",
 "target": {
        "@type": "AssetCollection",
        "source":  "http://example.com/media-catalogue",
        "refinement": [{
                "leftOperand": "runningTime",
                "operator": "lt",
                "rightOperand": {
                 "@value": "60",
                 "@type": "xsd:integer"
                 },
                "unit": "http://qudt.org/vocab/unit/
                   MinuteTime"
        }]
 },
 "action": "play"
}]
}
```

Listing 1.9: Constraint su asset: l'azione **play** è permessa solo sui target di durata strettamente inferiore a 60 minuti

```
{
"@context": "http://www.w3.org/ns/odrl.jsonld",
"@type": "Agreement",
"uid": "http://example.com/policy:4444",
"profile": "http://example.com/odrl:profile:12",
"permission": [{
  "target": "http://example.com/myPhotos:BdayParty",
```

```
  "assigner": "http://example.com/user44",
  "assignee": {
     "@type": "PartyCollection",
         "source":  "http://example.com/user44/friends",
         "refinement": [{
                "leftOperand": "foaf:age",
                "operator": "gt",
                "rightOperand": {
                 "@value": "17",
                 "@type": "xsd:integer"
                 }
         }]
        },
  "action": { "@id": "ex:view" }
}]
}
```

Listing 1.10: Constraint su party: l'azione **view** è permessa solo alle entità con età strettamente superiore a 17 anni

Rule

Come definito nel modello presente nella sezione 1.1.2, **rule** è una classe astratta che raccoglie gli aspetti comuni della classi **permission**, **prohibition**, and **duty**. Rappresenta una delle regole all'interno della policy. Presenta le seguenti proprietà:

- una **action**: azione regolamentata;
- una o nessuna **relation**: asset sul quale si applica la regola;
- una, nessuna o più **function**: funzioni che un party può avere all'interno di una regola;
- uno, nessuno o più **constraint** : limiti applicati alla validità della regola;
- uno o nessun identificativo univoco, necessario solo qualora si usare la regola per ereditarne le proprietà;

Le sottoclassi sono così definite:

- **permission**: permette un'azione sull'asset specificato, con tutti i **refinement** di quest'ultima soddisfatti; inoltre l'azione può essere eseguita solo se tutti i limiti

della regola sono soddisfatti e ogni dovere espresso come **duty** è stato rispettato. Un permesso rende obbligatoria la **relation** denominata **target**;

- **prohibition**: vieta un'azione sull'asset specificato, con tutti i **refinement** di quest'ultima soddisfatti; inoltre l'azione non può essere eseguita solo se tutti i limiti della regola sono soddisfatti; se si infrange il divieto, ogni dovere espresso come **remedy** deve essere eseguito. Un divieto rende obbligatoria la **relation** denominata **target**;

- **duty**: obbligo di eseguire un'azione, con tutti i **refinement** di quest'ultima soddisfatti; un dovere è compiuto se tutti i suoi limiti sono soddisfatti e la sua azione effettuata, con tutti i **refinement** definiti. Se un dovere non è stato compiuto, bisogna eseguirne le **consequences**, ovvero altri doveri da compiere.

Esempi di regole all'interno di policy:

```
{
 "@context": "http://www.w3.org/ns/odrl.jsonld",
 "@type": "Offer",
 "uid": "http://example.com/policy:9090",
 "profile": "http://example.com/odrl:profile:07",
 "permission": [{
    "target": "http://example.com/game:9090",
    "assigner": "http://example.com/org:xyz",
    "action": "play",
    "constraint": [{
        "leftOperand": "dateTime",
        "operator": "lteq",
        "rightOperand": { "@value": "2017-12-31", "@type"
           : "xsd:date" }
    }]
  }]
}
```

Listing 1.11: La regola esprime il permesso di eseguire l'azione **play** sul target fino al giorno 2017-12-31 compreso

```
{
 "@context": "http://www.w3.org/ns/odrl.jsonld",
 "@type": "Agreement",
 "uid": "http://example.com/policy:5555",
```

```
 "profile": "http://example.com/odrl:profile:08",
 "prohibition": [{
        "target": "http://example.com/photoAlbum:55",
        "action": "archive",
        "assigner": "http://example.com/MyPix:55",
        "assignee": "http://example.com/assignee:55"
 }]
}
```

Listing 1.12: La regola esprime il divieto di eseguire l'azione **archive** sul target

```
{
"@context": "http://www.w3.org/ns/odrl.jsonld",
"@type": "Agreement",
"uid": "http://example.com/policy:42",
"profile": "http://example.com/odrl:profile:09",
"obligation": [{
 "assigner": "http://example.com/org:43",
 "assignee": "http://example.com/person:44",
 "action": [{
   "rdf:value": {
     "@id": "odrl:compensate"
   },
   "refinement": [
     {
     "leftOperand": "payAmount",
     "operator": "eq",
     "rightOperand": { "@value": "500.00", "@type": "xsd:
        decimal" },
     "unit": "http://dbpedia.org/resource/Euro"
   }]
  }]
 }]
}
```

Listing 1.13: La regola esprime l'obbligo di eseguire l'azione **compensate**, specificando come **refinement** l'ammontare del pagamento

1.1.3 Problematica trattata

Gestione dei conflitti

La trattazione fatta fino ad ora per ODRL prende in considerazione la definizione di una singola policy per volta. Questo caso non rispecchia però le necessità reali che il linguaggio punta a soddisfare ed, in particolare, non rispecchia i casi d'uso definiti in MOSAICO, dove è naturale che vi sia un numero molto alto di policy definite.
ODRL propone già alcuni elementi per permettere la definizione di più policy in modo agevole, ad esempio:

- è possibile utilizzare la proprietà **inheritFrom** per permettere ad una policy di ereditare tutte le regole definite in un'altra policy;
- è possibile utilizzare la proprietà **conflict** definire una strategia di risoluzione dei conflitti; le strategia attualmente definite nel modello sono le seguenti:
  - **perm**: le regole di tipo **permission** hanno la priorità in caso di conflitto;
  - **prohibit**: le regole di tipo **prohibition** hanno la priorità in caso di conflitto;
  - **invalid**: in caso di conflitto, la policy risulta non valida nella sua interezza; è il valore di default se non definito esplicitamente.

Questo sistema di gestione dei conflitti risulta avere però una problematica fondamentale all'interno dei casi d'uso di MOSAICO: si riferisce a conflitti all'interno di una singola policy o che avvengono in seguito all'utilizzo della proprietà **inheritFrom**; risulta possibile notare che in uno scenario multi-owner, come quello di MOSAICO, l'utilizzo della proprietà **conflict** risulta inadatto, ad esempio:

1. si supponga che un ospedale, indicato come A, voglia rendere disponibili i propri dati di ricerca in un mercato come quello di MOSAICO;
2. risulta plausibile che la collezione di dati di questo ospedale venga inserita in una raccolta di referti medici, i cui owner sono vari ospedali situati in stati diversi;
3. il dato fornito da A viene accompagnato da una policy ODRL, la quale può utilizzare la proprietà **inheritFrom** per inserire nella propria policy le regole richieste da normative europee;
4. oltre a questo, A inserisce nella policy anche regole relative a norme sulla privacy vigenti nel proprio paese, settando la proprietà **conflict** a **prohibit**, al fine di proteggere i dati sensibili degli utenti trattati dalla propria collezione dati.

Se si considera lo scenario mostrato in esempio non multi-owner, ODRL riesce perfettamente ad soddisfare le esigenze dell'owner della collezione di dati, permettendogli di aderire alle normative europee ed, allo stesso tempo, di tutelare la privacy dei propri pazienti in conformità con le normative vigenti nel suo paese.
Nel caso in cui, la collezione di dati medici appartenga a più ospedali, questa soluzione non risulta più adatta, in particolare nel seguente caso:

1. un secondo ospedale, indicato come B, segue il medesimo procedimento ma lascia la proprietà **conflict** uguale ad **invalid**;
2. se le norme dei paesi di A e B non sono le stesse, entrambe le policy vengono invalidate nella loro totalità, anche se in disaccordo solo su un sottoinsieme di regole;
3. questo comportamento risulta deleterio sia per quanto concerne il mercato, poiché scoraggia degli owner a partecipare ad una collezione;
4. il comportamento mostrato risulta dannoso anche per quanto concerne i soggetti dei dati, non più tutelati da regole poiché invalidate.

Una possibile soluzione a questo problema è attuare il merging delle varie policy che targettano la collezione di dati, rendendo l'azione sulle varie regole più granulare; prendendo ad esempio l'ultimo scenario mostrato:

1. avendo due strategie di conflitto differenti, nessuna delle due viene considerata;
2. per ogni regola in conflitto, si tengono solamente le regole di divieto, seguendo l'obiettivo di MOSAICO di tutela della privacy;
3. per ogni regola non in conflitto, la regola viene mantenuta, seguendo l'obiettivo d MOSAICO di portare un dato di qualità ed accessibile;
4. per ogni regola definita come permessa solo in una policy, la si invalida, poiché la seconda policy non si esprime in merito.

Questo procedimento è solo uno dei possibili, in base alle casistiche di conflitto possibili e risulta in linea con i requisiti che MOSAICO punta a soddisfare, poiché a differenza di quanto definito in ODRL:

- porta ad una diminuzione della qualità del dato graduale, anziché ad una invalidazione totale della policy al primo conflitto;
- durante la diminuzione della qualità del dato, preserva comunque il maggior grado di tutela della privacy che le policy definite possono offrire, poiché permette sempre e solo azioni che hanno una regola esplicita che le riguarda.

Bibliography

[1] Chris Bizer. Linkeddata.

[2] Renato Iannell, Michael Steidl, Stuart Myles, and Víctor Rodríguez-Doncel. Odrl version 2.2 ontology.

[3] Renato Iannella, Michael Steidl, Stuart Myles, and Víctor Rodríguez-Doncel. Odrl vocabulary & expression 2.2.

[4] Renato Iannella and Serena Villata. Odrl information model 2.2.

[5] Benedict Whittam Smith and Víctor Rodríguez-Doncel. Odrl implementation best practices.