The Dublin Core Metadata Initiative (DCMI) [DCMI] provides a core metadata vocabulary, commonly referred to as Dublin Core. The original element set, from 1995, ~~consisted~~ contains ~~of~~ 15 broadly-defined elements ~~that are still available nowadays~~still in common use. ~~These~~ The core elements ~~are defined very broadly, in particular they~~ have no range specification. ~~, i.e., they can be used with a~~Arbitrary values can be used as objects. The core elements have been been expanded beyond the original fifteen. Existing elements have been refined and new elements have been added. ~~further refined and new types have been introduced.~~ This ~~more specific~~ expanded vocabulary is called ~~the terms~~ referred to as "DCMI Terms" and currently consists of 55 properties [DCTERMS].

The use of DCMI term is preferred. The Dublin Core element set has been deprecated~~The Dublin Core elements are considered legacy and the use of the DCMI terms is preferred~~. The two element sets have different namespaces. ~~They have different namespaces; if~~T ~~abbreviated, t~~he ~~elements~~ original element set is typically referred to with the *dc* prefix~~are usually used with the `dc` prefix~~, while ~~`dct` or~~ `dcterms` (or, dct) ~~prefix~~ is used as a prefix for the ~~terms~~newer DCMI element set.

DCMI terms hold a lot of provenance information and tell us about a resource, *when* it was affected in the past, *who* affected it and *how* it was affected. The other DCMI terms (description metadata), tell us *what* was affected. There is no direct information in Dublin Core describing *where* a resource was affected. Such information is usually only available for the publication of a resource (i.e., an action located at the address of the publisher). Note that spatial is not related to this question, as it is a descriptive property that links a resource to the location referred to in its content, but not to the location where it was created, modified, issued or published.

~~. Consider the following example for a metadata record:~~

~~Example 1: a simple metadata record:~~

```
ex:doc1 dct:title "A mapping from Dublin Core..." ;
    dct:creator ex:kai, ex:daniel, ex:simon, ex:michael ;
    dct:created "2012-02-28" ;
    dct:publisher ex:w3c ;
    dct:issued "2012-02-29" ;
    dct:subject ex:dublincore ;
    dct:replaces ex:doc2 ;
    dct:format "HTML" .
```

~~Clearly not all metadata statements deal with provenance. `dct:title`, `dct:subject` and `dct:format` are descriptions of the resource `ex:doc1`. They do not provide any information on how the resource was created or modified in the past. On the other hand, some statements imply provenance-related information. For example `dct:creator` implies that the document has been created and refers an author. Similarly, the existence of the `dct:issued` date implies that the document has been published. This information is redundantly implied by the `dct:publisher` statement as well. Finally, `dct:replaces`~~

relates the document to another document `ex:doc2` which had probably some kind of influence on `ex:doc1`.

Following this pattern, the existing metadata can be categorized as description metadata and provenance metadata. To be more precise, provenance metadata is defined as metadata providing provenance information according to the definition of the Provenance Working Group [PROV-DEF] and description metadata as all other metadata.

Based on this definition, the DCMI terms can be classified as follows:

Description metadata: abstract, accessRights, accrualMethod, accrualPeriodicity, accrualPolicy, alternative, audience, bibliographicCitation, conformsTo, coverage, description, educationLevel, extent, hasPart, isPartOf, format, identifier, instructionalMethod, isRequiredBy, language, mediator, medium, relation, requires, spatial, subject, tableOfContents, temporal, title, type.

Provenance metadata: available, contributor, created, creator, date, dateAccepted, dateCopyrighted, dateSubmitted, hasFormat, hasVersion, isFormatOf, isReferencedBy, isReplacedBy, issued, isVersionOf, license, modified, provenance, publisher, references, replaces, rightsHolder, rights, source, valid.

A This is a conservativemapping is provided below classification of provenance metadata. The mapping is by necessity somewhat conservative, as Iit can be argued that other elements placed in the description metadata set contain provenance information as well, depending on their their usage in a concrete implementation or application. .

According Based to on the proposed classificationthis, there are 25 (terms out of the 55 total) terms that can be considered as provenance related. These terms can further be further categorized according to the question they answer regarding the provenance of a resource:

Date and Time Terms (When?)

This category contains date and time related terms. Dates typically belong to the provenance record of a resource. It can be questioned whether a resource changes by being published or not. Depending on the application, however, the publication can be seen as an action that changes the state of the resource. Two dates can be considered special regarding their relevance for provenance: available and valid. They are different from the other dates as by definition they can represent a date range. Often, the range of availability or validity of a resource is inherent to the resource and known beforehand – consider the validity of a passport or the availability of a limited special offer published on the web. In these cases, there is no action involved that makes the resource invalid or unavailable, it is simply determined by the validity range. On the other hand, if an action is involved, e.g., a resource is declared invalid because a mistake has been found, then it is relevant for its provenance.

Agency Terms (Who? ~~(*contributor, creator, publisher, rightsHolder*):~~)
This category contains agent related terms. ~~Category that includes a~~All properties that have `dct:Agent` as range, i.e., a resource that acts or has the power to act. The contributor, creator, and publisher clearly influence the resource and therefore are important for its origin. This is not immediately clear for the `dct:rightsHolder`, but as ownership is considered the important provenance information for many resources, like artworks, it is included in this category.

~~**When? (*available, created, date, dateAccepted, dateCopyrighted, dateSubmitted, issued, modified, valid*): Dates typically belong to the provenance record of a resource. It can be questioned whether a resource changes by being published or not. Depending on the application, however, the publication can be seen as an action that changes the state of the resource. Two dates can be considered special regarding their relevance for provenance: available and valid. They are different from the other dates as by definition they can represent a date range. Often, the range of availability or validity of a resource is inherent to the resource and known beforehand – consider the validity of a passport or the availability of a limited special offer published on the web. In these cases, there is no action involved that makes the resource invalid or unavailable, it is simply determined by the validity range. On the other hand, if an action is involved, e.g., a resource is declared invalid because a mistake has been found, then it is relevant for its provenance.~~**

Derivation Terms (How?~~? (*isVersionOf, hasVersion, isFormatOf, hasFormat, references, isReferencedBy, replaces, isReplacedBy, source, rights, license*)~~)
This category contains derivation related terms. ~~:~~Resources are often derived from other resources. In this case, the original resource becomes part of the provenance record of the derived resource. Derivations can be further classified as `dct:isVersionOf`, `dct:isFormatOf`, `dct:replaces`, `dct:source`. `dct:references` is a weaker relation, but it can be assumed that a referenced resource influenced the described resource and therefore it is relevant for its provenance. The respective inverse properties do not necessarily contribute to the provenance of the described resource, e.g., a resource is usually not directly affected by being referenced or by being used as a source – at most indirectly, as the validity state can change if a resource is replaced by a new version. However, inverse properties belong to the provenance related terms as they can be used to describe the relations between the resources involved. Finally, licensing and rights are considered part of the provenance of the resource as well, since they restrict how the resource has been used by its owners.

Table 1 summarizes the terms in their respective categories:

Table 1: Categorization of the Dublin Core Terms

Category	Sub-category	Terms
Descriptive metadata	-	abstract, accessRights, accrualMethod, accrualPeriodicity, accrualPolicy, alternative, audience, bibliographicCitation, conformsTo, coverage, description, educationLevel, extent,

		hasPart, isPartOf, format, identifier, instructionalMethod, isRequiredBy, language, mediator, medium, relation, requires, spatial, subject, tableOfContents, temporal, title, type
Provenance	Who	contributor, creator, publisher, rightsHolder
Provenance	When	available, created, date, dateAccepted, dateCopyrighted, dateSubmitted, issued, modified, valid
Provenance	How	isVersionOf, hasVersion, isFormatOf, hasFormat, license, references, isReferencedBy, replaces, isReplacedBy, rights, source

This leaves one very special term: *provenance*. This term is defined as a "statement of any changes in ownership and custody of the resource since its creation that are significant for its authenticity, integrity, and interpretation" [DC-TERMS], which corresponds to the traditional definition of provenance for artworks. Despite being relevant for provenance, this definition may overlap partially with almost half of the DCMI terms, which specify concrete aspects of provenance of a resource.

~~In summary, the DCMI terms – and therefore any Dublin Core metadata record – hold a lot of provenance information and tell us about a resource, *when* it was affected in the past, *who* affected it and *how* it was affected. The other DCMI terms (description metadata), tell us *what* was affected. There is no direct information in Dublin Core describing *where* a resource was affected. Such information is usually only available for the publication of a resource (i.e., an action located at the address of the publisher). Note that spatial is not related to this question, as it is a descriptive property that links a resource to the location referred to in its content, but not to the location where it was created, modified, issued or published.~~ Consider the following example for a metadata record:

Example 1: a simple metadata record:

```
 ex:doc1 dct:title "A mapping from Dublin Core..." ;
     dct:creator ex:kai, ex:daniel, ex:simon, ex:michael ;
     dct:created "2012-02-28" ;
     dct:publisher ex:w3c ;
     dct:issued "2012-02-29" ;
     dct:subject ex:dublincore ;
     dct:replaces ex:doc2 ;
     dct:format "HTML" .
```

In the example above, `dct:title`, `dct:subject` and `dct:format` are descriptions of the resource `ex:doc1`. These terms do not provide any information on how the resource was created or modified in the past. Other statements in the set do imply provenance-related information. For example `dct:creator` implies that the document has been created and refers an author. Similarly, the existence of the `dct:issued` date implies that the document has been published. This information is redundantly implied by the `dct:publisher` statement as well. Finally, `dct:replaces` relates the document to another document `ex:doc2` which had probably some kind of influence on `ex:doc1`.

1.1 Namespaces

The namespaces used through the document can be seen in Table 2 below:

Table 2: Namespaces used in the document

owl	<http://www.w3.org/2002/07/owl#>
rdfs	<http://www.w3.org/2000/01/rdf-schema#>
prov	<http://www.w3.org/ns/prov#>
dct	<http://purl.org/dc/terms/>

2. Mapping from Dublin Core to PROV

A mapping between Dublin Core Terms and PROV-O has many advantages. First, it can provide valuable insights into the different characteristics of both data models (in particular it explains PROV from a Dublin Core point of view). Second, such a mapping can be used to extract PROV data from the ~~huge~~ large amount of Dublin Core data ~~that is~~ available on the Web today. Third, ~~it~~ the mapping can translate PROV data to Dublin Core and make it accessible for applications that understand Dublin Core. ~~Last, but not least~~And finally, ~~it~~ the mapping can lower the barrier to entry ~~to~~ for ~~adopt~~ PROV adoption. ~~, as s~~Simple Dublin Core statements can be used as starting point ~~to generate~~ for PROV data generation.

2.1 Basic considerations

Substantially, a complete mapping from Dublin Core to PROV consists of three parts:

1) **Direct mappings** between terms that can be expressed in form of subclass or subproperty relationships in RDFS – or equivalent relationships in OWL.

2) Definition of new **refinements** (subclasses or subproperties) of the target vocabulary to reflect the expressiveness of the source vocabulary.

3) Provision of **complex mappings** that create statements in the target vocabulary based on statements in the source vocabulary. Since the mapping produces blank nodes for each `dct` statement, a clean-up phase with strategies for reducing the blank nodes is also necessary.

2.2 What is ex:doc1? Entities in Dublin Core

Consider the example metadata record shown at the beginning of this document (in example 1). As a `dc` metadata record describes the resulting document as a whole, it is not clear how this document relates to the different states that the document had until it reached its final state. For example, a document may have a `dct:created` date and a `dct:issued` date. According to the PROV ontology, the activity of issuing a document involves two different states of the document: the document before it was issued and the

issued document. Each of these states correspond to a different specialization of the document, even if the document has not changed. Generally, there are two possibilities to deal with this issue:

1) Create new instances of entities, typically as blank nodes, that are all related to the original document by means of `prov:specializationOf`. This leads to bloated and not very intuitive data models, e.g. think about the translation of a single `dct:publisher` statement, where anyone would expect to somehow find some activity and agent that are directly related to the document (as in Figure 1).

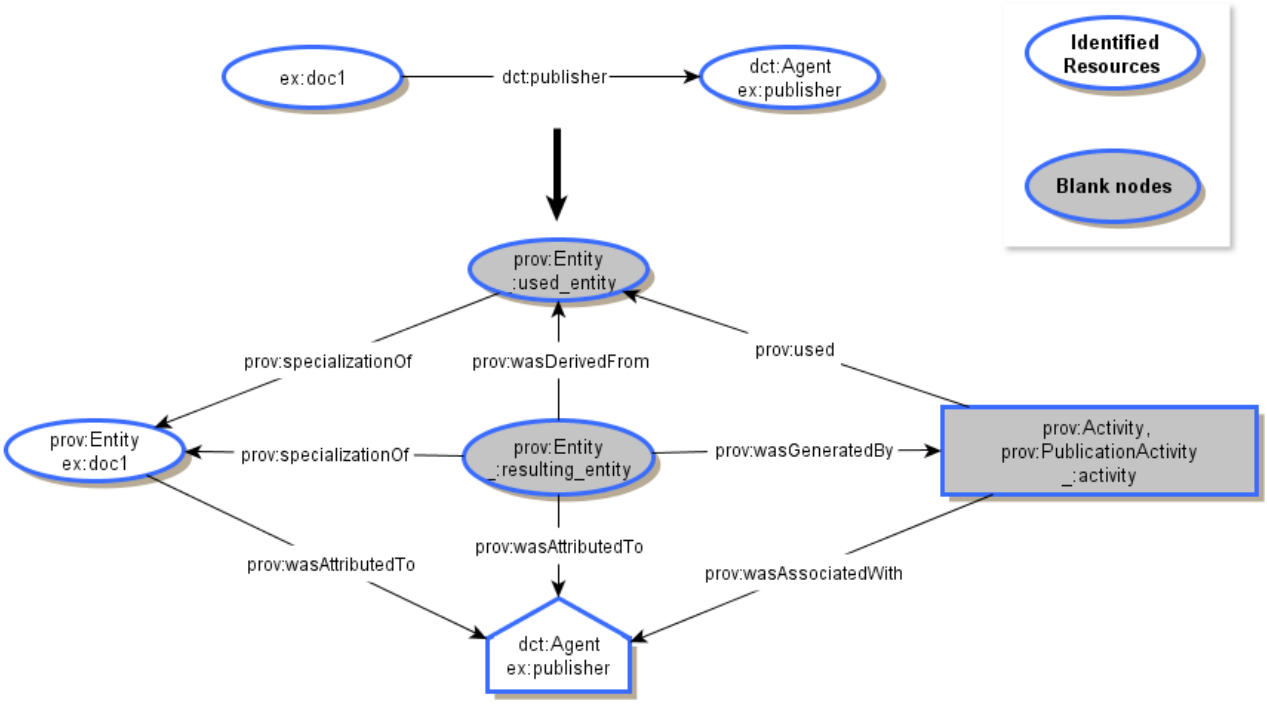

Figure 1. A mapping example creating blank nodes for each state of the resource. In PROV entities are represented with ellipses, activities with rectangles and agents with pentagons.

2) Use the original resource (`ex:doc1`) as the instance used and generated as `prov:Entity`. However, to have an activity that uses an entity and generates the same entity or to have different activities that generate the same entity at different points in time is not compliant with the PROV constraints [PROV-CONTRAINTS]. Figure 2 provides an example that illustrates this approach.

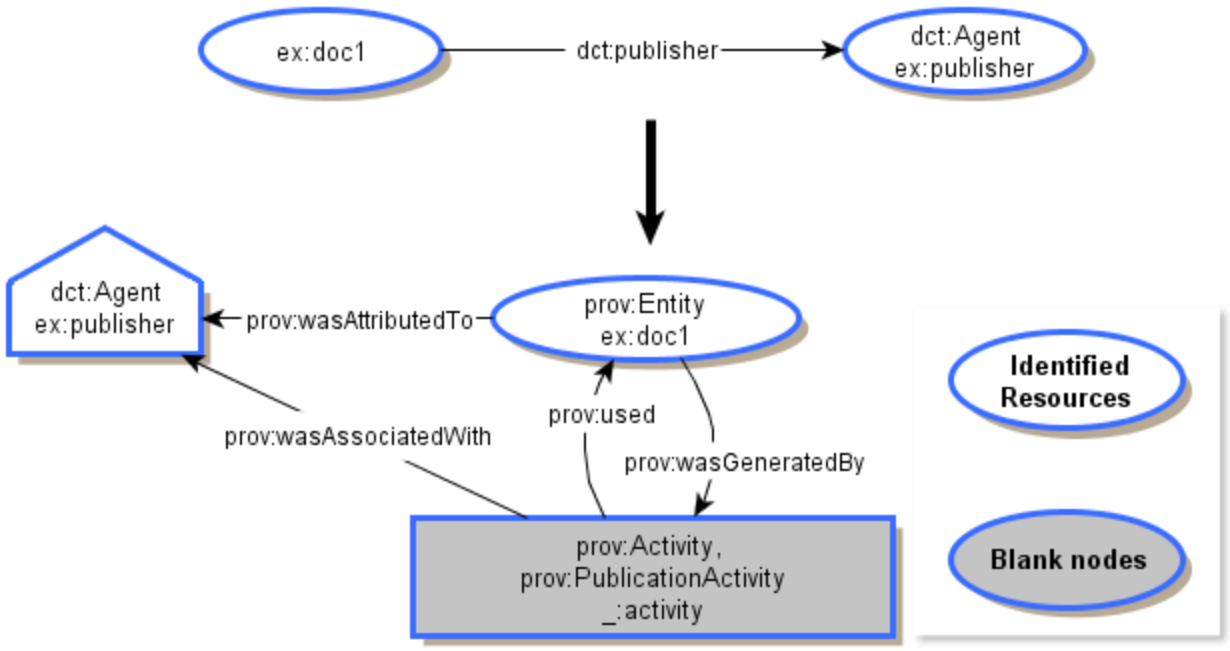

Figure 2. A mapping example conflating blank nodes in the same resource. The used and generated resources have the same identifier.

Since the first option is the most conservative with respect to the underlying semantics, it has been chosen as guideline in the complex mapping. Blank nodes are used for the mapping, although any naming mechanism could be provided if necessary, leaving the conflating of nodes to the clean-up phase.

2.3 Direct mappings

The direct mappings provide basic interoperability using the integration mechanisms of RDF. By means of OWL 2 RL reasoning, any PROV application can at least make some sense from Dublin Core data. The direct mappings also contribute to the formal definition of the vocabularies by translating them to PROV.

Dublin Core, while less complex from a modeling perspective, is more specific about the type of the activity taking place. PROV provides general attribution, and the details about the kind of influence that an activity or an agent had are left to custom refinements of the PROV classes and properties.

Table 3 and Table 4 provide the detailed mapping plus the rationale for each term. The rest of the terms can be found in the list of terms left out of the mapping.

Table 3: Direct mappings

DC Term	Relation	PROV Term	Rationale
dct:Agent	owl:equivalentClass	prov:Agent.	Both `dct:Agent` and `prov:Agent` refer to

			the same concept: a resource that has the power to act (which then has responsibility for an activity).
dct:rightsHolder	rdfs:subPropertyOf	prov:wasAttributedTo	The rights holder has the attribution of the activity that created the licensed resource.
dct:creator	rdfs:subPropertyOf	prov:wasAttributedTo	A creator is the agent who created the resource. He is the one involved in the creation activity that led to the resource. He has the attribution for that activity
dct:publisher	rdfs:subPropertyOf	prov:wasAttributedTo	A publisher has the attribution of the publishing activity that led to the published resource.
dct:contributor	rdfs:subPropertyOf	prov:wasAttributedTo	A contributor is involved either in the creation activity or in the updating of the resource. Therefore he/she is attributed to take part in those activities.
dct:isVersionOf	rdfs:subPropertyOf	prov:wasDerivedFrom	`dct:isVersionOf` refers to "a related resource to which the current resource is a version, edition or adaptation". Hence the current resource has been derived from the original one.
dct:isFormatOf	rdfs:subPropertyOf	prov:alternateOf	`dct:isFormatOf` refers to another resource which is the same but in another format. Thus the mapping is straightforward to `prov:alternateOf`.

dct:hasFormat	rdfs:subPropertyOf	prov:alternateOf	See rationale for `dct:isFormatOf`
dct:replaces	rdfs:subPropertyOf	prov:wasInfluencedBy	This mapping is not straightforward. There is a relation between two resources when the former replaces the latter, but it is not necessarily derivation, revision, specification or alternate. Thus, the term is mapped to `prov:wasInfluencedBy`
dct:source	rdfs:subPropertyOf	prov:wasDerivedFrom	In Dublin Core, `dct:source` is defined as a "related resource from which the described resource is derived", which matches the notion of derivation in PROV-DM ("a transformation of an entity in another")
dct:type	owl:equivalentProperty	prov:type	Both properties relate two resources in a similar way: the nature of the resource (or genre).
dct:created	rdfs:subPropertyOf	prov:generatedAtTime	`dct:created` is a property used to describe the time of creation of an entity, which corresponds to the time of its generation. The rationale to map this property as a subclass of `prov:generatedAtTime` is that resources in Dublin Core may have many dates associated to them (creation, modification, issue, etc.), each of which could correspond to a

			different version of the document. In this case, the creation is the first date asserted to the document, but doesn't necessarily correspond to the current version of the resource.
dct:issued	rdfs:subPropertyOf	prov:generatedAtTime	Date when the resource was issued. It is mapped as a subproperty of `prov:generatedAtTime` because the issued resource is an entity itself, which has been generated at a certain time.
dct:dateAccepted	rdfs:subPropertyOf	prov:generatedAtTime	The rationale is similar to the previous two properties: the version of the resource which was accepted could be different from the created or issued one.
dct:dateCopyrighted	rdfs:subPropertyOf	prov:generatedAtTime	See `dct:dateAccepted`
dct:dateSubmitted	rdfs:subPropertyOf	prov:generatedAtTime	See `dct:dateAccepted`
dct:modified	rdfs:subPropertyOf	prov:generatedAtTime	See `dct:dateAccepted`

With the direct mapping, a metadata record such as example 1 will infer that the resource was `prov:generatedAtTime` at two different times. Although this may seem inconsistent, it is supported by PROV and it is due to the difference between Dublin Core and PROV resources: while the former conflates more than one version or "state" of the resource in a single entity, the latter proposes to separate all of them. Thus, the mapping produces provenance that complies with the current definition of entity but it does not comply with all the PROV constraints [PROV_CONSTRAINTS].

Some properties have been found to be superproperties of certain prov concepts. These can be seen below in Table 4:

Table 4: Direct mappings (2)

PROV Term	Relation	DC Term	Rationale
prov:hadPrimarySource	rdfs:subPropertyOf	**dct:source**	The definition of `prov:hadPrimarySource`

prov:wasRevisionOf	rdfs:subPropertyOf	**dct:isVersionOf**	("something produced by some agent with direct experience and knowledge about the topic") is more restrictive than `dct:source` ("A related resource from which the described resource is derived"). Similar to the previous property, `prov:wasRevisionOf` is more restrictive in the sense that it refers to revised version of a resource, while `dct:isVersionOf` involves versions, editions or adaptations of the original resource.

Table 5 enumerates the mapping of the `dct` properties that map to inverse relationships in PROV. These have been separated in a different table because they don't belong to the core of PROV.

Table 5: Direct mappings to the PROV terms not included in the core

PROV Term	Relation	DC Term	Rationale
dct:hasVersion	rdfs:subPropertyOf	prov:hadDerivation	Inverse property of `dct:isVersionOf`.
dct:isReplacedBy	rdfs:subPropertyOf	prov:influenced	Inverse property of `dct:replaces`

2.4 PROV refinements

To properly reflect the meaning of the Dublin Core terms, more specific subclasses are needed:

```
 prov:PublicationActivity      rdfs:subClassOf     prov:Activity .
 prov:ContributionActivity     rdfs:subClassOf     prov:Activity .
 prov:CreationActivity         rdfs:subClassOf     prov:Activity,
prov:ContributionActivity .
 prov:ModificationActivity     rdfs:subClassOf     prov:Activity .
 prov:AcceptanceActivity       rdfs:subClassOf     prov:Activity .
 prov:CopyrightingActivity     rdfs:subClassOf     prov:Activity .
 prov:SubmissionActivity       rdfs:subClassOf     prov:Activity .
 prov:PublisherRole            rdfs:subClassOf     prov:Role .
 prov:ContributorRole          rdfs:subClassOf     prov:Role .
 prov:CreatorRole              rdfs:subClassOf     prov:Role,
prov:ContributorRole .
```

Custom refinements of the properties should be omitted as they would be identical to the Dublin Core terms. If these more specific properties are needed, the Dublin Core terms should be used directly, according to the direct mappings presented in section 2.3.

2.5 Complex Mappings

The complex mappings consist on a set of patterns defined to generate qualified PROV statements from Dublin Core statements. This type of qualification may not be always needed, and it is the choice of the implementor whether to use them or not depending on the use case. It is also important to note that not all the direct mappings have a complex mapping associated, just those which imply a specific activity: creation, publication, etc. The complex mappings are provided in form of SPARQL CONSTRUCT queries, i.e., queries that describe a resulting RDF graph based on another RDF graph found in the original data. We divide the queries in different categories:

2.5.1 Entity-Agent mappings (Who)

In this category, we have three terms: `dct:contributor`, `dct:creator` and `dct:publisher`. The three of them can be mapped with the same pattern, similar to the one presented in Figure 1. The only changes required are the roles and activities involved for each term.

In the text below, variables `?document` and `?agent` are set to different matching values depending on the data found in the triple store. The graph in the CONSTRUCT part can be seen as a template where the variables are placeholders that are filled with the values found in the data. The mapping corresponds to the graph in Figure 1 (with small changes for creator and rightsHolder). With this mapping, the difference in the complexity becomes obvious. Many blank nodes are created, so a subsequent clean-up phase that relates them and provides stable URIs for the entities is required. Depending on the implementation, URIs can also be coined here for every specialization. The implementation proposed in this document is an example that works conservatively. The assumption is that no further information about the identity of the specializations is available.

2.5.1.1 dct:creator

A creator is the agent associated with role CreatorRole in the CreationActivity that created a specialization of the entity (?document).

```
CONSTRUCT {
  ?document a prov:Entity ;
            prov:wasAttributedTo ?agent.

  ?agent a prov:Agent .

  _:activity a prov:Activity, prov:CreationActivity ;
            prov:wasAssociatedWith ?agent;
            prov:qualifiedAssociation [
                    a prov:Association;
```

```
                        prov:agent ?agent;
                        prov:hadRole prov:CreatorRole .
                ].

    _:resulting_entity a prov:Entity ;
                prov:specializationOf ?document ;
                prov:wasGeneratedBy _:activity ;
                prov:wasAttributedTo ?agent.

} WHERE {
    ?document dct:creator ?agent.
}
```

2.5.1.2 dct:contributor

Contributor is mapped following the previous pattern. Only the roles and activities change:

```
CONSTRUCT {
    ?document a prov:Entity ;
                prov:wasAttributedTo ?agent .

    ?agent a prov:Agent .

    _:activity a prov:Activity, prov:ContributionActivity ;
                prov:wasAssociatedWith ?agent ;
                prov:qualifiedAssociation [
                        a prov:Association ;
                        prov:agent ?agent ;
                        prov:hadRole prov:ContributorRole .
                ].

    _:resulting_entity a prov:Entity ;
                prov:specializationOf ?document ;
                prov:wasGeneratedBy _:activity ;
                prov:wasAttributedTo ?agent .

} WHERE {
    ?document dct:contributor ?agent .
}
```

2.5.1.3 dct:publisher

In case of publication, a second specialization representing the entity before the publication is necessary:

```
  CONSTRUCT {
    ?document a prov:Entity ;
                prov:wasAttributedTo ?agent .

    ?agent a prov:Agent .

    _:used_entity a prov:Entity;
                prov:specializationOf ?document.

    _:activity a prov:Activity, prov:PublicationActivity ;
                prov:used _:used_entity;
```

```
                prov:wasAssociatedWith ?agent ;
                prov:qualifiedAssociation [
                        a prov:Association ;
                        prov:agent ?agent ;
                        prov:hadRole prov:PublisherRole .
                ].

    _:resulting_entity a prov:Entity ;
                prov:specializationOf ?document ;
                prov:wasDerivedFrom _:used_entity
                prov:wasGeneratedBy _:activity ;
                prov:wasAttributedTo ?agent .

} WHERE {
    ?document dct:publisher ?agent .
}
```

2.5.2 Entity-Date mappings (When)

Dates often correspond with a who-property, e.g., creator and created or publisher and issued. Therefore, they lead to similar complex patterns (providing a date instead of an agent associated with the corresponding activity). When using Dublin Core terms, it is usual to see that a resource is annotated with several `dct` assertions like creator, publisher, issued, date, etc., but in this phase of the mapping each term is treated independently.

2.5.2.1 dct:created

```
CONSTRUCT{
?document a  prov:Entity .

_:activity a prov:Activity, prov:CreationActivity ;

# The "output"
_:created_entity a prov:Entity ;
            prov:specializationOf ?document ;
            prov:wasGeneratedBy _:activity ;
            prov:wasGeneratedAtTime ?date;
            prov:qualifiedGeneration [
                a prov:Generation ;
                prov:atTime ?date  ;
                prov:activity _:activity .
            ] .
} WHERE {
 ?document dct:created ?date.
}
```

2.5.2.2 dct:issued

```
CONSTRUCT{
?document a prov:Entity .

_:activity a prov:Activity, prov:PublicationActivity ;
```

```
            prov:used _:used_entity .

# The "input"
 _:used_entity a prov:Entity .
            prov:specializationOf ?document .

 # The "output"
 _:iss_entity a prov:Entity ;
            prov:specializationOf ?document ;
            prov:wasGeneratedBy _:activity ;
            prov:wasGeneratedAtTime ?date;
            prov:wasDerivedFrom _:used_entity ;
            prov:qualifiedGeneration [
                 a prov:Generation ;
                 prov:atTime ?date  ;
                 prov:activity _:activity .
            ] .
 } WHERE {
  ?document dct:issued ?date.
 }
```

2.5.2.3 dct:modified

```
 CONSTRUCT{
 ?document a prov:Entity .

 _:activity a prov:Activity, prov:ModificationActivity ;
            prov:used _:used_entity .

# The "input"
 _:used_entity a prov:Entity .
            prov:specializationOf ?document .

 # The "output"
 _:modified_entity a prov:Entity ;
            prov:specializationOf ?document ;
            prov:wasGeneratedBy _:activity ;
            prov:wasGeneratedAtTime ?date;
            prov:wasDerivedFrom _:used_entity ;
            prov:qualifiedGeneration  [
                 a prov:Generation ;
                 prov:atTime ?date  ;
                 prov:activity _:activity .
            ] .
 } WHERE {
  ?document dct:modified ?date.
 }
```

2.5.2.4 dct:dateAccepted

```
 CONSTRUCT{
 ?document a prov:Entity .

 _:activity a prov:Activity, prov:AcceptanceActivity ;
```

```
            prov:used _:used_entity .

# The "input"
 _:used_entity a prov:Entity .
            prov:specializationOf ?document .

 # The "output"
 _:accepted_entity a prov:Entity ;
            prov:specializationOf ?document ;
            prov:wasGeneratedBy _:activity ;
            prov:wasGeneratedAtTime   ?date;
            prov:wasDerivedFrom       _:used_entity ;
            prov:qualifiedGeneration  [
                 a prov:Generation ;
                 prov:atTime ?date  ;
                 prov:activity _:activity .
            ] .
 } WHERE {
  ?document dct:dateAccepted ?date.
 }
```

2.5.2.5 dct:dateCopyrighted

```
 CONSTRUCT{
 ?document a prov:Entity .

 _:activity a prov:Activity, prov:CopyrightingActivity ;
            prov:used _:used_entity .

# The "input"
 _:used_entity a prov:Entity .
            prov:specializationOf ?document .

 # The "output"
 _:copyrighted_entity a prov:Entity ;
            prov:specializationOf ?document ;
            prov:wasGeneratedBy _:activity ;
            prov:wasGeneratedAtTime ?date;
            prov:wasDerivedFrom _:used_entity ;
            prov:qualifiedGeneration [
                 a prov:Generation ;
                 prov:atTime ?date  ;
                 prov:activity _:activity .
            ] .
 } WHERE {
  ?document dct:dateCopyrighted ?date.
 }
```

2.5.2.6 dct:dateSubmitted

```
 CONSTRUCT{
 ?document a prov:Entity .

 _:activity a prov:Activity, prov:SubmissionActivity ;
            prov:used _:used_entity .
```

```
# The "input"
 _:used_entity a prov:Entity .
            prov:specializationOf ?document .

 # The "output"
 _:submitted_entity a prov:Entity ;
            prov:specializationOf ?document ;
            prov:wasGeneratedBy _:activity ;
            prov:wasGeneratedAtTime ?date;
            prov:wasDerivedFrom _:used_entity ;
            prov:qualifiedGeneration  [
                 a prov:Generation ;
                 prov:atTime ?date  ;
                 prov:activity _:activity .
            ] .
 } WHERE {
  ?document dct:dateSubmitted ?date.
 }
```

2.5.3 Cleanup

The clean-up phase depends on how implementers interpret the described resources. The approach presented in this document is conservative and it leads to the proliferation of blank nodes. Blank nodes could be renamed to specific identifiers by the implementer, in order to avoid obtaining additional blank nodes when reapplying the construct queries presented in the previous section.

Providing a set of rules to conflate the blank nodes is not in the scope of this document. However, the group has created a list of suggestions for implementers with proposals on how this could be achieved:

1) **Conflate properties referring to the same state of the resource**: In Dublin Core certain properties complement each other (e.g., creator and created, publisher and issued, modified and contributor, etc.). By combining some of the queries, some of the records could be grouped creating more complete PROV assertions.

The example below shows how to conflate the blank nodes for `dct:creator` and `dct:created` properties:

```
 CONSTRUCT{
 ?document a prov:Entity .

 _:activity a prov:Activity, prov:CreationActivity.
            prov:wasAssociatedWith ?agent
            prov:qualifiedAssociation [
                 a prov:Association;
                 prov:agent ?agent;
                 prov:hadRole prov:CreatorRole .
            ] .

 # The "output"
 _:created_entity a prov:Entity ;
```

```
            prov:specializationOf ?document ;
            prov:wasGeneratedBy _:activity ;
            prov:wasGeneratedAtTime   ?date;
            prov:qualifiedGeneration  [
                 a prov:Generation ;
                 prov:atTime ?date  ;
                 prov:activity _:activity .
        ] .
} WHERE {
 ?document dct:creator  ?agent;
           dct:created  ?date.
}
```

Figure 3 shows a graphical representation of the pattern:

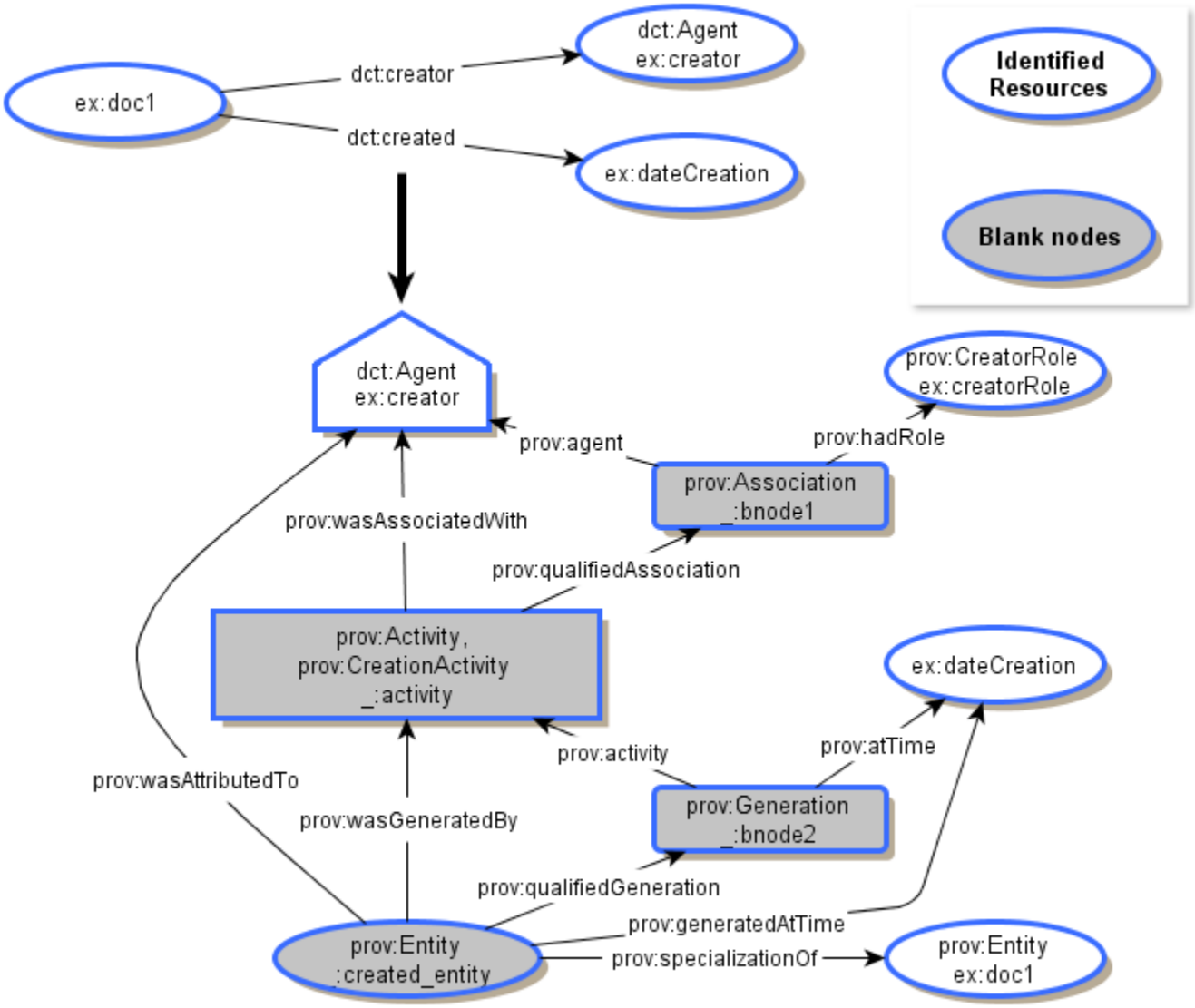

Figure 3. Gathering complementing properties to conflate blank nodes.

2) Another solution is to **sort all the activities according to their date**, if known, and conflate the blank nodes result of one activity and the input of the subsequent activity, in case they are both specializations of the same entity. Figure 4 shows a graphical example with two different activities (creation and publication) that happened at different points in time. Instead of creating different blank nodes for the respective usage and generation, both activities share the same blank node (`_:created_entity`).

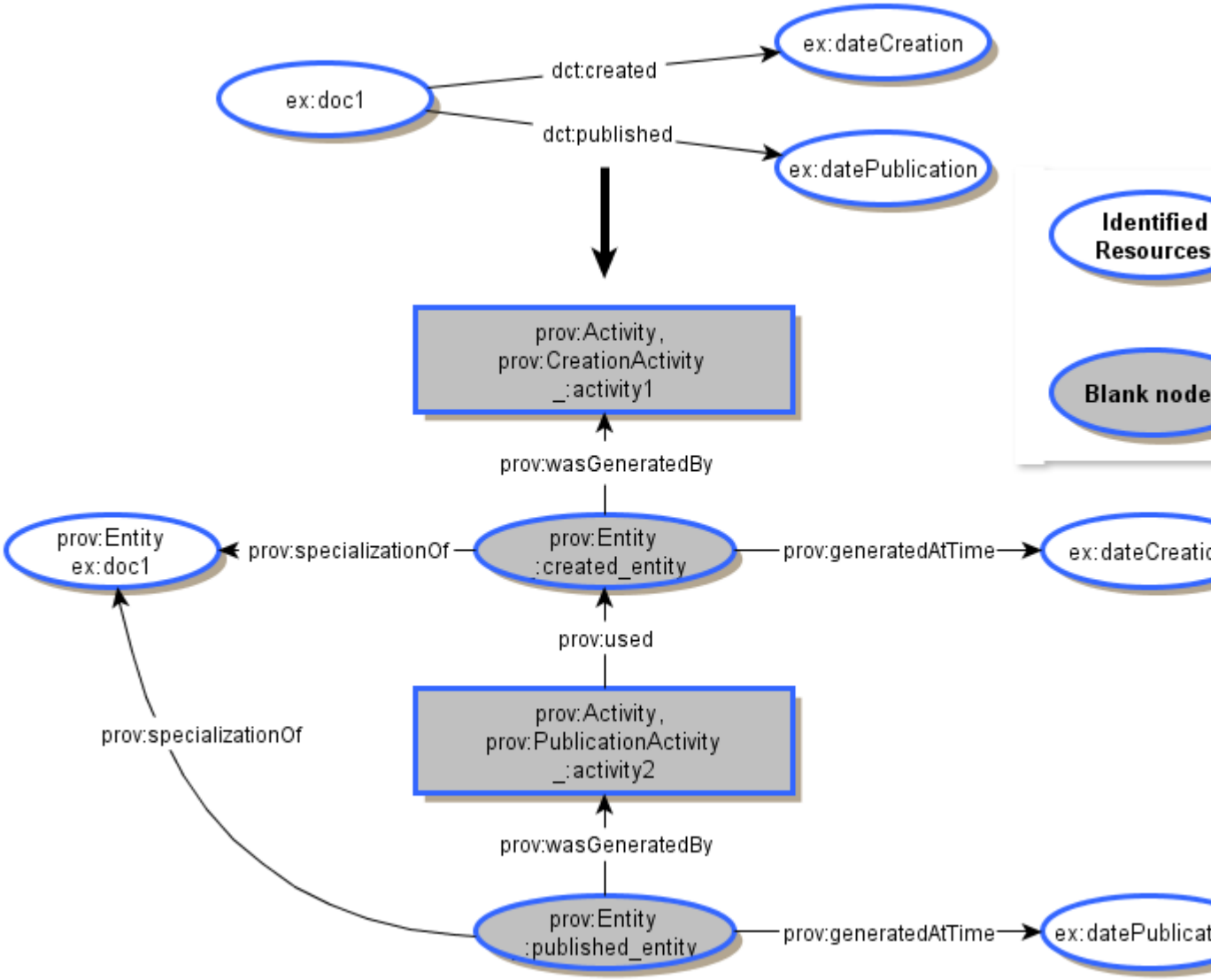

Figure 4. Sorting the activities by date to conflate blank nodes.

3) Finally, another solution is to **ignore all the specializations of `ex:doc1` and use the resource itself**. This solution would avoid the majority of the blank nodes, linking all the activities with the resource. However, the results would be confusing in case there are several Dublin Core statements describing the same resource (like `dct:publisher` and `creator`), since most of the activities would use and generate the same resource at different times (all the provenance of the different versions of the resource would be

conflated in the same entity). A graphical representation of an example can be seen in Figure 2.

2.6. List of terms excluded from the mapping

Table 6: List of terms excluded from the mapping

Term	Category	Rationale
dct:abstract	Descriptive metadata	Summary of the resource. Thus, not part of its provenance.
dct:accrualMethod	Descriptive Metadata	Method by which items are added to a collection. It doesn't describe the action itself, so it is out of the scope of the mapping
dct:accrualPeriodicity	Descriptive metadata	Frequency of the addition of items to a collection.
dct:accrualPolicy	Descriptive metadata	Policy associated with the insertion of items to a collection. It could be used to enrich the qualified involvement, but there is no direct mapping of this relationship.
dct:alternative	Descriptive metadata	Refers to an alternative name of the resource.
dct:audience	Descriptive metadata	The audience for whom the resource is useful.
dct:conformsTo	Descriptive metadata	Indicates the standard to which the resource conforms to (if any).
dct:coverage	Descriptive metadata	Topic of the resource.
dct:description	Descriptive metadata	An account of the resource.
dct:educationLevel	Descriptive metadata	The educational level of the audience for which the resource is intended too.
dct:extent	Descriptive metadata	Size or duration of the resource.
dct:format	Descriptive metadata	Format of the resource.
dct:identifier	Descriptive metadata	An unambiguous reference on a given context.
dct:instructionalMethod	Descriptive metadata	Method used to create the knowledge that the resource is supposed to support.
dct:isPartOf	Descriptive metadata	Inverse of `dct:hasPart`.
dct:isRequiredBy	Descriptive metadata	The current resource is required for supporting the function of another resource. This is not related the provenance, since it refers to something that may not have happened yet

		(e.g., a library dependency, but the program that needs it hasn't been executed yet).
dct:language	Descriptive metadata	Language of the resource.
dct:mediator	Descriptive metadata	Entity that mediates access to the resource.
dct:medium	Descriptive metadata	Material of the resource.
dct:requires	Descriptive metadata	Inverse property of `dct:isRequiredBy` (see `dct:isRequiredBy`).
dct:hasPart	Descriptive metadata	A resource that is included in the current resource. Since entity composition is out of the scope of PROV, this property has been excluded from the mapping
dct:spatial	Descriptive metadata	Spatial characteristics of the content of the resource (e.g., the book is about Spain). Thus it can't be mapped to `prov:hadLocation`.
dct:subject	Descriptive metadata	Subject of the resource.
dct:tableOfContents	Descriptive metadata	List of subunits of the resource.
dct:temporal	Descriptive metadata	Temporal characteristics of which the resource refers to (e.g., a book about 15th century).
dct:title	Descriptive metadata	Title of the resource.
dct:type	Descriptive metadata	Type of the resource.
dct:bibliographicCitation	Descriptive metadata	Property that relates the literal representing the bibliographic citation of the resource to the actual resource (e.g., `:el_Quijote dct:bibliographicCitation "Miguel de Cervantes Saavedra: El Quijote, España"`).
dct:references	Provenance: How	This term could be used to refer to sources that have been used to create the document, but it could be also used to cite the sources that are not relevant for the current work. For this reason it has been dropped from the mapping.
dct:isRefrencedBy	Provenance: How	Inverse to `dct:references`.
dct:accessRights	Provenance: How	Agents who can access the resource (security status). Since the privileges of the resource are part of the description of the resource, the property has been excluded from the mapping.
dct:license	Provenance:	License of the resource. It has been left out of

	How	the mapping because there is no term in PROV-O to represent this information.
dct:rights	Provenance: How	Metadata about the rights of the resource.
dct:date	Provenance: When	Date is a very general property. It is the superproperty which all the other specialize, but there is no equivalent concept in PROV. It has been excluded from the mapping
dct:available	Provenance: When	Property that states when a resource is available. The group could not reach consensus on how to map this property, so it was finally dropped from the mapping.
dct:valid	Provenance: When	Property that states when a resource is valid. The notion of invalidation is defined in PROV-DM, but not the notion of validation. Thus this property is left out of the mapping.
dct:relation	Provenance	A related resource. This relationship is very broad and could relate either provenance resources or not. Therefore it could be seen as a superproperty of `prov:wasDerivedFrom`, `prov:wasInfluencedBy`, `prov:alternateOf`, `prov:specializationOf`, etc. Thus there is no direct mapping.

3. Mapping from PROV to DC

The mapping from PROV to Dublin Core is not part of this note. It can be questioned, if a mapping without additional information would provide meaningful data. If refinements are used, the mapping would be straight forward using the inverse of the mapping patterns used in this document. However, without such refinements, few Dublin Core statements can be inferred, apart from some unqualified dates. Dublin Core includes provenance information, but the focus lies on the description of the resources. Pure PROV data models a provenance chain, but it contains almost no information about the resulting resource itself.

A. Acknowledgements

We would like to thank Antoine Isaac, Ivan Herman, Timothy Lebo, Simon Miles and Satya Sahoo for their comments and feedback, and Idafen Santana Pérez for his help with the HTML generation.

B. References

B.1 Normative references

No normative references.

B.2 Informative references

[DCMI]
Dublin Core Metadata Initiative. URL: http://dublincore.org/

[DCTERMS]
Dublin Core Terms Vocabulary. URL: http://dublincore.org/documents/dcmi-terms/

[PROV-CONTRAINTS]
James Cheney, Paolo Missier, and Luc Moreau (eds.) *Constraints of the PROV Data Model.*. 2011. W3C Working Draft. URL: http://www.w3.org/TR/prov-constraints/

[PROV-DM]
Luc Moreau, Paolo Missier *The PROV Data Model and Abstract Syntax Notation*. 15 December 2011. W3C Working Draft. (Work in progress.) URL: http://www.w3.org/TR/2011/WD-prov-dm-20111215/

[PROV-DEF]
W3C Provenance Working Group's Definition of Provenance.URL: http://www.w3.org/TR/2012/WD-prov-dm-20120724/#dfn-provenance

[PROV-O]
Timothy Lebo, Satya Sahoo, Deborah McGuinness *The PROV Ontology: Model and Formal Semantics*. 13 December 2011. W3C Working Draft. (Work in progress.) URL: http://www.w3.org/TR/2011/WD-prov-o-20111213