Metadata Application Profile Provenance with Extensible Authoring Format and PAV Ontology

Nishad Thalhath[1](✉), Mitsuharu Nagamori[2], Tetsuo Sakaguchi[2], and Shigeo Sugimoto[2]

[1] Graduate School of Library, Information and Media Studies, University of Tsukuba, Tsukuba, Japan
nishad@slis.tsukuba.ac.jp

[2] Faculty of Library, Information and Media Studies, University of Tsukuba, Tsukuba, Japan
{nagamori,saka,sugimoto}@slis.tsukuba.ac.jp,
https://www.slis.tsukuba.ac.jp

Abstract. Metadata application profiles (MAP) serve a critical role in the of metadata interoperability. Singapore framework recommends publishing the application profiles as documentation, with detailed usage guidelines aimed to maximize reusability and interoperability. Authoring, maintenance, versioning, and ensuring the availability of previous versions along with changelogs are vital steps involved in MAP publishing. The longevity of the schema is a critical part of metadata longevity. MAP should provide sufficient administrative information and versioning to ensure the provenance and longevity as a record of changes of the metadata instance. The authors propose to include actionable changelogs and provenance information within an extensible MAP authoring format. The proposal also includes a recommendation on MAP versioning and publishing with PAV, a lightweight ontology for Provenance, Authoring, and Versioning.

Keywords: Metadata · Metadata schema · Application profiles · Schema versioning · Semantic web · Linked data · PAV Ontology

1 Introduction

Metadata application profiles (MAP) are data element schemas from various namespaces mixed and customized for a specific application [9]. MAPs are the best mechanism to express consensus of any metadata instance by documenting the elements, policies, guidelines, and vocabularies for that particular implementation along with the schemas, and applicable constraints. Application profiles also provide the term usage specifications and support interoperability by representing domain consensus, alignment, and the structure of the data [1,10].

Provenance information is vital for application profiles. Changelogs of application profile versions help to ensure the metadata instance's longevity. The longevity of the schema is essential for metadata longevity. Metadata schema provenance should be documented and maintained for the preservation of metadata [13]. Application profile should provide sufficient administrative information, such as creator, date of release, version, and usage rights. Versioning of the application profile is crucial as it is a record of the application profile as well as metadata changes. Keeping changelogs might help to migrate data-sets to new profiles or create crosswalks to upgrade the instances. For Linked Open Data (LOD), changelogs help to update linked datasets as well.

Through this paper, the authors are attempting to:

1. Extend and clarify a previously proposed [25] extensible authoring format [24] to include structured and actionable changeset with a notion that the source of the MAP can include an actionable timeline of its development.
2. Use a lightweight ontology to distinguish and point the source of the MAP as well as the published MAP resources [4].
3. Use the same ontology to notate the provenance information with identifiable roles on authoring, publishing, and contributing for collaborative MAP development.

The anticipated outcomes of these proposals are:

1. Distinguish the source of the application profile from the published versions to baseline the concepts of authoring formats and expression formats for application profiles.
2. Identifying and retrieving application profiles and its versions, including changelogs, can be automated with the help of semantic linking of MAP resources.
3. A source of MAP with an interoperable authoring format consists of an actionable timeline can help to maintain the longevity of the schema. Declared roles of contribution can act as a means of provenance for MAP resources.

1.1 Application Profile Expression Formats

Dublin Core Metadata Initiative (DCMI) defines one of the earliest guidelines to express application profiles, which can be in various formats, as Description Set Profiles (DSP). DSP is a constraint language for Dublin Core Application Profiles (DCAP) based on the Singapore framework for application profiles [18]. XML or RDF can be used as an expression format for DSP.

Singapore framework recommends publishing the application profiles in human- readable expression formats as a documentation, with detailed usage guidelines aimed to maximize reusability and interoperability. Expressing application profile in human readable formats require much more components than textual descriptions of first-order elements such as properties and classes. As a result, the expression of an application profile in human readable formats is expected to have schematic representation, changelogs as well as detailed administrative information.

For the machine-actionable expressions, other than the XML and RDF, new standards are emerging and being widely accepted. Evolution of Linked Data encourages to express the application profiles in semantic web friendly formats like JSON-LD and OWL. Considering the developments in data linking and reuse, compelling use cases for expressing application profiles in promising data validation formats like ShEx or SHACL is increasing. Including these futuristic expression formats in application profile publishing will expand the scope of its usage as well as assures long-term usability.

1.2 Current Status and Availability of Application Profiles

Application profiles are not standardized in terms of availability, maintenance, and distribution. It requires human involvement to identify MAPs [15]. Because of this manual effort, curating and archiving MAPs is difficult and costly. In addition to automated methods, numerous registry initiatives also rely on manual contributions. Most of the application profiles are available only in human-friendly formats, and to distinguish them from other types of documents; this requires human involvement in the identification process. It is challenging to extract structured application profile data from spreadsheets or PDF documents. Lack of versioning, changes logs, and access to previous versions have a substantial impact on metadata information's longevity and provenance. The absence of unified publication formats limits the automated processing of application profiles, thereby limiting the number of application profiles accessible in various attempts to register and curate them. The limited number of application profiles also restricts the primary purpose of metadata registries in using application profiles to promote interoperability and reuse [17]. There is also a lack of a standardized way to link data to the MAP it is based on [21,22].

1.3 Challenges in Application Profile Development

To develop and manage application profiles, there are recommendations such as Me4DCAP which provides a set of guidelines to define, construct, and validate MAPs [14]. However, authoring tools and formats which are dedicated to MAP is less in number. Usually, application profile maintainers have to use different tools to create different expression formats, and this makes the whole process tedious for most of the domain experts. As a result, a large number of application profiles were authored and published only in the human-readable document formats. Availability of previous versions is not ensured and most of the MAPs doesn't maintain older versions in a publicly accessible format.

Different communities have different levels of experience in the technical aspects of application profile expression formats. There is a severe lack of guidance for developing and publishing metadata application profiles. The barrier is the limited number of well-defined samples and initiatives for archiving and curating application profiles. For creating application profiles, there are not many well-accepted authoring formats or pre-processors.

1.4 Yet Another Metadata Application Profile (YAMA) as an Application Profile Authoring Format

Source formats used for application profile publication can be considered as an authoring format. This source formats can be processed with the help of processing tools such as a format converter or a parsing system to generate different expression formats of that application profile. A format to author the application profile cannot be considered as an expression of the application profile in all situations if the format is not a standard expression. The expression capabilities of such formats are dependable to its processors or conversion tools. This clear separation between authoring and expression formats is illustrated in Fig. 1.

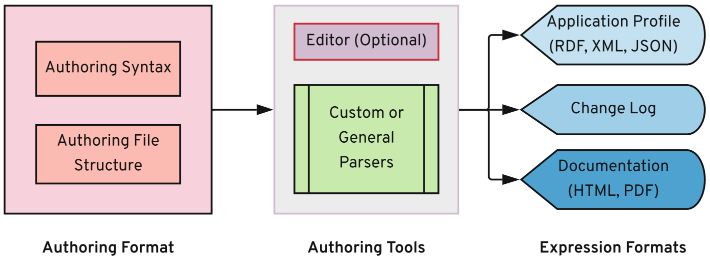

Fig. 1. Authoring formats and expression formats for application profiles

For application profile, authors proposes Yet Another Metadata Application Profile (YAMA) as an extensible authoring format to address shortcomings of previous proposals [23]. Despite extensive knowledge of MAP, YAMA is meant to be simple enough that domain experts can use it. YAMA uses YAML Ain't Markup Language (YAML), a robust human-friendly data serialization format with various implementations in most popular programming languages and considered to be JSON's superset [2]. Basic structure of YAMA MAP section is explained in Fig. 2.

YAMA is also an attempt to resolve the lack of a workflow in authoring metadata application profiles. Given the increasing popularity of workflows based on GitHub, different output formats, and extensibility to various proposals such as ShEx, DCAT, PROV removes the need for repetitive tasks in the maintenance of metadata application profiles. YAMA is an intermediate MAP format to produce or convert different standard application profile expression formats.

YAMA is extensible with custom elements and structure. For example, custom elements can be added to the document tree, as per the demand of the use case. The only restriction is that custom elements cannot be from reserved element sets. Capabilities of YAMA could be extended without any large-scale

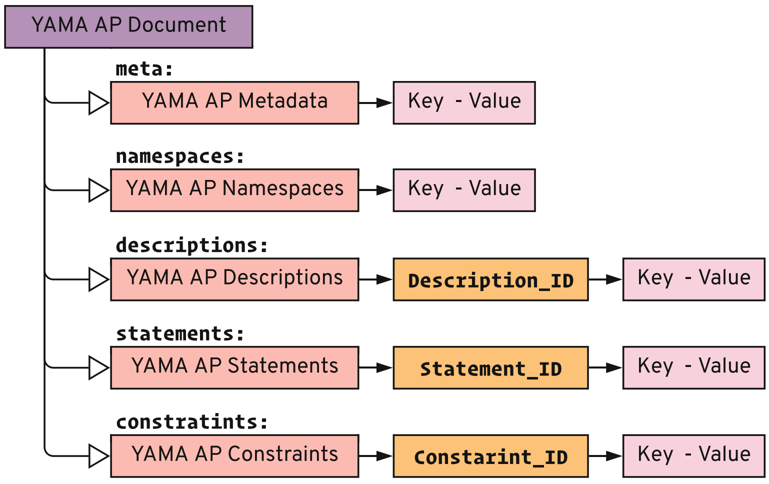

Fig. 2. Structure of YAMA MAP

implementation changes within the scope of YAML specification. YAMA is based on DC-DSP, and a minimal DC-DSP is mandatory to express a MAP in YAMA. YAMA also includes a structured syntax to record modifications of a YAMA document named as change-sets, in addition to extensible key-values and structure. YAMA change-sets can be used to record changes of a MAP over any other versions. Change-sets are adapted from RFC 6902 JavaScript Object Notation (JSON) Patch [19], with the changes marked as an action to a path. Every change use 'status' as a reserved value to indicate status changes like 'deprecation.' This extensible nature of YAMA documents is explained in Fig. 3.

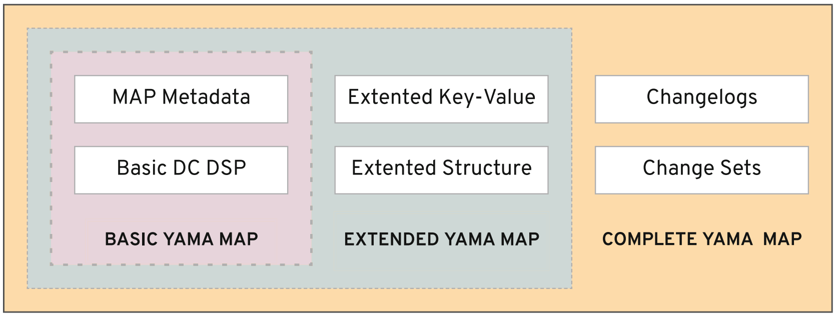

Fig. 3. Extensibility of a YAMA application profile

2 Related Work

As an application format, DCMI proposed a constrained language for Dublin Core Application Profiles named Description Set Profile (DSP). As an authoring format for DSP, a MoinMoin wiki syntax was introduced to embed Application Profiles in web pages. Later, Simple DSP (SDSP) [7]. A simplified form of DSP using spreadsheets as an authoring format was developed as part of the Metabridge project [17]. Recently, the DCMI application profile Special Interest Group is working on improving DSP [6]. Library of Congress BIBFRAME project developed a web-based editor for BibFrame Profiles [5]. Linked Data for Production 2 (LD4P2) project modified and released BIBFRAME editor for general application profile creation named Sinopia Profile Editor [11].

There is no extensibility of all these stated authoring formats. A format's extensibility is critical to its acceptance, which helps different communities to adopt a simple base format and introduce specific domain requirements. It will also help to create different standard formats from the same source document without relying on the common elements. The authors previously proposed an extensible authoring format named Yet Another Metadata Application Profile (YAMA) [25] using YAML[1] syntax and validated its extensibility over existing similar proposals [24].

Li and Sugimoto proposed a provenance model named DSP-PROV [13] to keep track of structural changes of metadata schemas. The DSP-PROV model applies PROV to the Dublin Core Application Profile. Different from the above proposal, this paper is treating application profile documents as a digital resource and attempting to use a lightweight ontology to map different versions of the published MAP and its provenance.

3 Methodology

The authors are attempting to extend a previously proposed MAP authoring format with an actionable timeline [23]. With the consideration that the format is to be a complete source of MAP authoring and versioning, a lightweight ontology is introduced to notate the authoring and versioning of MAP. The ontology is introduced with a notion that it can express different versions of the MAP as well as stakeholders and authoring source of the MAP.

3.1 Actionable Changesets as Timeline of MAP

YAMA is extended with two different sets of change mapping options. An actionable change record named 'changesets' - a collection of changes declared using a custom adaptation of JSON-PATCH - along with minimal metadata for the set of changes. Changesets are declared within the 'changes' path of the YAMA document. JSON patch is originally intended to use as HTTP-PATCH method for

[1] https://yaml.org.

JavaScript Object Notation (JSON) [RFC4627][2] - a standard format for storing and exchanging structured data. HTTP PATCH [RFC5789][3] method extends the Hypertext Transfer Protocol (HTTP) [RFC2616][4] as a method to perform partial modifications to resources. A simple JSON patch is shown in Fig. 4.

Adaption of JSON-Patch as a possible means of recording changes within the application profile authoring environment helps to makes the changes actionable without any lock-in as JSON-Patch is widely adapted and there are plenty of implementations in every popular programming languages. This acceptance helps the implementors to keep the format open for independent development and tooling within the workflow of MAP development.

A JSON Patch consists of sequential operations applied to a JSON object with one operation (op) element. As per RFC6902, valid operations are - add, remove, replace, move, copy, and test. Each operation must declare one path element which is a JSON Pointer - defined as per RFC6901 - points to a location to modify within the given JSON document. A JSON Pointer composed of a string of tokens separated by '/' characters. These tokens can be a specific key in objects or indexes of arrays.

The remaining part of a JSON Patch operation consists of more elements depends on the specific type of operation.

```
{
    "op": "remove",
    "path": "/statements/statement_id/"
}
```

Fig. 4. A basic JSON-Patch object indicating a removal operation

In theory, a YAMA document is a constrained YAML expression of a MAP which can be abstracted or converted into a valid JSON structure. The JSON patch is applied to this JSON structure instead of the YAML document. In order to make the JSON patch actionable for generating the pre or post-change versions of a MAP, authors extended the JSON patch by including a new optional elements `previous_value` which is applicable only for remove and replace operations. Another proposed additional element in the context of an application profile is `status` - which can notate the changes in status, such as deprecation, proposed, reserved, and obsolete. An example changeset expressed within YAMA is shows in Fig. 5. Minimal mandatory metadata elemets for YAMA changeset is given in Table 1.

Along with changesets, YAMA is extended with an optional changelog section, which is a human-readable list of changes with minimal metadata.

[2] https://tools.ietf.org/html/rfc6902.
[3] https://tools.ietf.org/html/rfc5789.
[4] https://tools.ietf.org/html/rfc2616.

```yaml
# YAMA
changes:
  cs_20181108_01:
    version: 1.2
    previous_version: 1.1
    date: 2018-11-08
    changeset:
      ch_20181108_01:
        op: replace # remove, add, replace, copy, test
        path: /statements/pr_type/max
        value: n
        previous_value: 0
```

Fig. 5. Example of YAMA ChangeSet

Table 1. Metadata for the changeset

Element	Usage
`version`	Version of the MAP after the change
`previous_version`	Version of the MAP, to which the change is applied
`date`	Date of change in ISO 8601 (not the date of release)

Changelogs are not meant to be actionable but act as a structured collection of human-readable descriptions of changes, which can be changes intended to be documented but does not have any impact on the structure of the MAP document. Also, this section can serve as an alternate but meaningful textual representation of changes instead of utilizing the changeset. This changelog section is proposed for authors prefer to utilize another means of change management, such as a version control systems, or authors with minimal technical expertise on creating an actionable JSON-patch. A schematic representation of YAMA's provenance components and their outcome is expressed in Fig. 6.

3.2 PAV Ontology as a Means of MAP Provenance

Provenance, Authoring and Versioning ontology (PAV)[5] [4] is developed as a lightweight ontology for notating minimal information which is essential for documenting the provenance, authoring, and versioning of resources published in the web. PAV clearly distinguishes between contributors, authors and publishes of digital resources. PAV is capable of representing the provenance of originating resources that have been accessed, transformed, and consumed.

PAV utilizes the W3C provenance ontology PROV-O, in order to describe authoring, publishing, and digital maintenance of online resources. PAV does not define any explicit classes, domain, or ranges; instead, every property is meant to be directly used in describing an online resource. This direct usage minimalizes

[5] http://purl.org/pav.

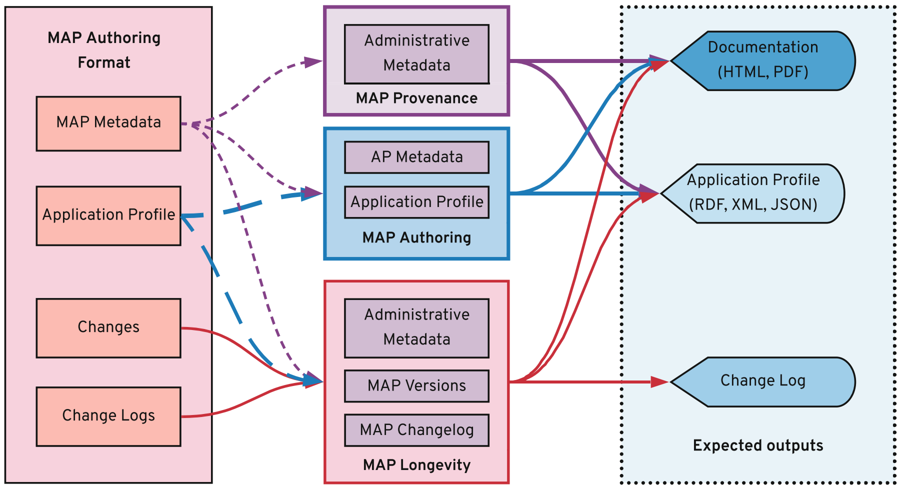

Fig. 6. YAMA with actionable changesets and changelogs mapped to their expected outputs

the efforts required for expressing resources using an ontology. Being lightweight over PROV-O is the main reason for considering PAV to be a means of expressing MAP resources [4].

There are vocabularies similar to PAV such as Dublin Core Terms (DC Terms) [3], PROV-O [12], OPM [16], and Provenance Vocabulary [8]. Among that PROV-O is the most suitable and previously considered in many other studies to express MAP provenance. PROV-O is similar to a generic framework for describing provenance in a different range of applications. However, using PROV-O alone may not be suitable in expressing necessary details for the specific provenance involving authoring and versioning. PAV can be considered as a specialization of PROV-O by facilitating more straightforward relationships for expressing common provenance for digital resources in the web [4]. PROV-O implements terms useful in tracing the origin of a resource, its derivations, and the relationship between these different resources. PROV-O is also capable of expressing the different entities contributed to the resource. In short, PROV-O can be considered as a general provenance data model extendable for domain-specific provenance information. For example, PROV-O does not distinguish between authors, editors, and contributors - which is a noticeable distinction in use-cases like collaborative MAP authoring and publishing based on public repositories such as GitHub.

PAV based framework is proposed in the context of MAP authoring and publishing with these intentions.

1. Identify the persons and organizations or agents involved in the application profile development. Also, distinguish their roles as contributor or creator of the published MAP.

Table 2. Subset of PAV authoring properties mapped to YAMA MAP metadata elements

YAMA	PAV element	Description
`creator`	`pav:authoredBy`	The author of the MAP (person or agent)
`creator`	`pav:createdBy`	The author of the MAP (person or agent)
`publisher`	`pav:curatedBy`	The Publisher, generally the organisation
`contributors`	`pav:contributedBy`	The collaborative agents, such as people who are not a part of the authors but contributed through GitHub etc
`date`	`pav:authoredOn`	The date of authoring the MAP
`date`	`pav:curatedOn`	The date of publishing the MAP
`date`	`pav:contributedOn`	Same as the date of publishing

Table 3. Subset of PAV provenance properties mapped to YAMA MAP metadata elements

YAMA metadata	PAV element	Description
`creator`	`pav:createdBy`	The author of the MAP (person or agent)
`publisher`	`pav:createdWith`	The tool, software or authoring format of the MAP. eg YAMA, Sinopia
`date`	`pav:createdOn`	The date of authoring the MAP
`URL`	`pav:retrievedFrom`	The published URL of the MAP
`source`	`pav:importedFrom`	The source of the MAP, eg: GitHub repository URL, Google Docs URL etc

Table 4. Subset of PAV versioning properties mapped to YAMA MAP metadata elements

YAMA metadata	PAV element	Description
`version`	`pav:version`	The version identifier of the MAP. A semantic version (SemVer) is recommended
`previous_version`	`pav:previousVersion`	Previous version identifier of the MAP. Current version is assumed to be a derived from this version
`based_on`	`pav:derivedFrom`	The base schema that the application profile is derived from. Example, DCAP, BibFrame, SDSP etc.
`date`	`pav:lastUpdatedOn`	The last updated date is expected, but this update is meant for changes that doesn't break the MAP structure, such as fixing a spelling
`version`	`pav:hasVersion`	MAP has accessible versions
`version`	`pav:hasCurrentVersion`	Current version of the MAP
`version`	`pav:hasEarlierVersion`	Earlier versions of the MAP

2. Mapping of MAP versions, release, and updates by distinguishes between published and last modified dates.
3. Track and distinguish the versions and source of the MAP, such as differentiating the provenance for the published versions of the application profiles and source repositories, the version control systems or authoring environment.

A detailed schematic explanation MAP versioning expression with PAV is narrated in Fig. 7. Tables 2, 3 and 4 shows the possible mapping of YAMA metadata elements a subset of PAV ontology.

4 Validation

To validate the proposal, a popular public application profile, The DCAT Application profile for data portals in Europe (DCAT-AP) can be used. DCAT-AP an application profile based on W3C's Data Catalogue vocabulary (DCAT). DCAT is implemented for describing public sector datasets in Europe to enable a cross-data portal search for open data sets and make them searchable. DCAT-AP is published in Joinup portal[6], but the sources are maintained in a GitHub repository[7]. DCAT-AP repository does not use any authoring format or preprocessors but maintains and releases the MAP in individual expression formats.

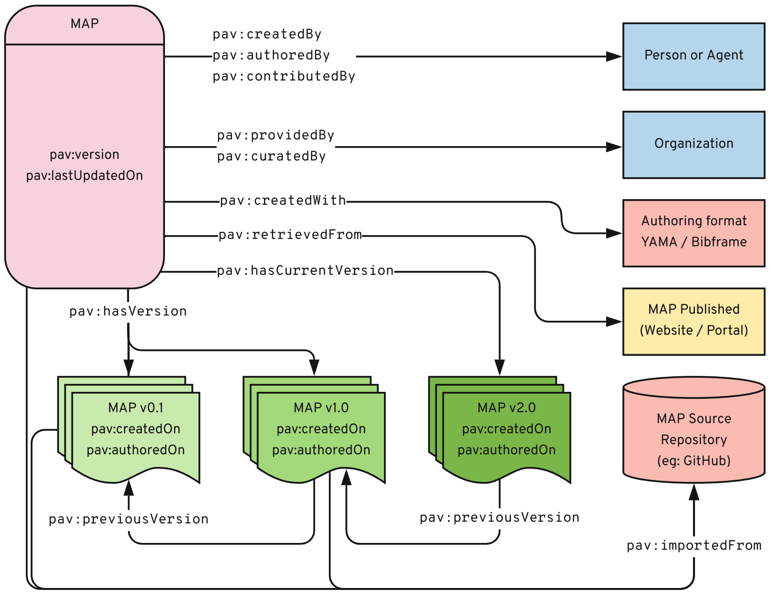

Fig. 7. MAP publication is expressed in PAV ontology

[6] https://joinup.ec.europa.eu/solution/dcat-application-profile-data-portals-europe.
[7] https://github.com/SEMICeu/DCAT-AP.

As a well-maintained MAP, the repository holds three different versions - v1.1, v1.2, and v1.2.1. RDF expression of the MAP points to the previous version, but the whole versioning is not mapped within the RDF [20]. A minimal expression of DCAT-AP provenance with PAV in RDF is demonstrated below.

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix pav: <http://purl.org/pav/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

<https://joinup.ec.europa.eu/solution/dcat-application-
    profile-data-portals-europe>
   pav:createdBy  [
    foaf:name "DCAT-AP Working Group" ;
    foaf:homepage <https://joinup.ec.europa.eu/node/64331>
    ] ;
   pav:authoredBy [
    foaf:name "Makx Dekkers" ;
    foaf:homepage <http://makxdekkers.com/>
    ], [
    foaf:name "Vassilios Peristeras" ;
    foaf:homepage <http://www.deri.ie/users/vassilios-
    peristeras/>
    ], [
    foaf:Name "Nikolaos Loutas" ;
    foaf:homepage <http://nikosloutas.com/>
    ];
   pav:curatedBy <https://joinup.ec.europa.eu/> ;
   pav:providedBy [
    foaf:name "European Commission" ;
    foaf:homepage <http://ec.europa.eu/>
    ] ;

   # GitHub contributors, need not to be authors or
    editors
   pav:contributedBy <https://github.com/SEMICeu/DCAT-AP/
    graphs/contributors>;

   pav:version "1.2.1"^^xsd:string;
   pav:hasCurrentVersion <https://joinup.ec.europa.eu/
    release/dcat-ap/121>;
   pav:previousVersion <https://joinup.ec.europa.eu/
    release/dcat-ap/12>;
   pav:hasErlierVersion <https://joinup.ec.europa.eu/
    release/dcat-ap/11>;
   pav:hasVersion <https://joinup.ec.europa.eu/release/
    dcat-ap/121>;
   pav:hasVersion <https://joinup.ec.europa.eu/release/
    dcat-ap/12>;
   pav:hasVersion <https://joinup.ec.europa.eu/release/
    dcat-ap/11>;
   pav:hasVersion <https://joinup.ec.europa.eu/node
    /69559>;
   pav:wasDerivedFrom <https://www.w3.org/TR/vocab-dcat
    -2/>;
   # GitHub main repository
   pav:importedFrom <https://github.com/SEMICeu/DCAT-AP>.

<https://joinup.ec.europa.eu/release/dcat-ap/121>
    pav:version "1.2.1"^^xsd:string;
    pav:createdOn "2019-05-28"^^xsd:date;
    pav:authoredOn "2019-05-28"^^xsd:date;
    pav:importedOn "2019-05-28"^^xsd:date;
    # GitHub repository versioned branch
    pav:wasDerivedFrom <https://github.com/SEMICeu/DCAT-AP
    /tree/1.2.1>;
    # GitHub repository draft version branch
    pav:sourceAccessedAt <https://github.com/SEMICeu/DCAT-
    AP/tree/1.2.1-draft>;
    pav:previousVersion <https://joinup.ec.europa.eu/
    release/dcat-ap/12>;
    pav:hasErlierVersion <https://joinup.ec.europa.eu/
    release/dcat-ap/11>.

<https://joinup.ec.europa.eu/release/dcat-ap/12>
    pav:version "1.2"^^xsd:string;
    pav:authoredOn "2018-11-08"^^xsd:date;
    pav:wasDerivedFrom <https://github.com/SEMICeu/DCAT-AP
    /tree/1.2>;
    pav:importedFrom <https://github.com/SEMICeu/DCAT-AP/
    tree/1.2-draft>;
    pav:previousVersion <https://joinup.ec.europa.eu/
    release/dcat-ap/11>;
    pav:hasErlierVersion <https://joinup.ec.europa.eu/node
    /69559>.

<https://joinup.ec.europa.eu/release/dcat-ap/11>
    pav:version "1.1"^^xsd:string;
    pav:authoredOn "2016-06-08"^^xsd:date;
    pav:wasDerivedFrom <https://github.com/SEMICeu/DCAT-AP
    /tree/1.1>;
    pav:previousVersion <https://joinup.ec.europa.eu/node
    /69559>.

<https://joinup.ec.europa.eu/node/69559>
    pav:version "1"^^xsd:string.
```

5 Limitations and Future Work

As an authoring format, YAMA can be extended to include the actionable changesets and parsable changelog. And PAV ontology can be used to point the source of the MAP, in which the YAMA changeset can be exposed as the timeline of the application profile. The main limitation of this approach is its inability in pointing to a standard format of the actionable changeset. A processor or system capable of understanding YAMA's YAML format as well as JSON-Patch is required to parse the changeset and develop the timeline of the application profile from it. So it is recommended that the authors or publishes tender required efforts to properly expose the changesets in other standard actionable formats as well. Even though YAML and JSON-Patch are comparatively more uncomplicated concepts for structured data, they demand the authors to have the skill sets and capabilities to deal with these formats. Mainly these formats need to be generated or modified using a 'real text editor' as there is not yet any known dedicated graphical editor implementation for YAMA.

PAV ontology is capable enough to point to versions and sources of the application profiles. The authors made this recommendation purely on the notion that MAPs are published as a package of expression formats and documentation. PAV is not directly usable in differentiating these formats within the application profile package or even pointing to individual format. For example, PAV may not be sufficient enough in distinguishing and pointing to the individual files representing the human-readable documentation or machine-actionable expressions like RDF and ShEx. Also, PAV mapping needs to implemented in templates or generators, used in producing expression formats from YAMA. Webpages liked to the application profiles requires to use RDFa or JSON-LD to include the ontology in expressing the versions and source with PAV.

The future work is to adopt ontologies to cover YAMA changesets with the capability of mentioning changes within an actionable and semantic approach. Notating the relation between individual expressions formats inside the publishable application profile package is also being investigated.

6 Conclusion

Providing a simplified authoring format can substantially promote the application profile creation efforts. Utilizing extensibility of this authoring format to include actionable changelog as the timeline of MAP creation can help in ensuring longevity. The authors are attempted to explain the possibility of a previously proposed extensible authoring format for application profiles with an advanced changeset. This paper also demonstrates adopting a lightweight ontology to notate the versioning of this application profiles with distinguishing its source from the published expressions. Any attempts to ensure the provenance and longevity of the metadata application profile will also help to ensure the provenance of the schema. Schema maintenance will help to achieve better goals in data interoperability and seamless linking of data with automated techniques.

Acknowledgement. This work was supported by JSPS KAKENHI Grant Number JP18K11984.

References

1. Baca, M.: Introduction to Metadata, July 2016. http://www.getty.edu/publications/intrometadata
2. Ben-Kiki, O., Evans, C., döt Net, I.: YAML Ain't Markup Language (YAML™) Version 1.2, October 2009. https://yaml.org/spec/1.2/spec.html
3. Board, D.U.: DCMI: DCMI Metadata Terms. https://www.dublincore.org/specifications/dublin-core/dcmi-terms/2012-06-14/
4. Ciccarese, P., Soiland-Reyes, S., Belhajjame, K., Gray, A.J., Goble, C., Clark, T.: PAV ontology: provenance, authoring and versioning. J. Biomed. Semant. **4**(1), 37 (2013). https://doi.org/10.1186/2041-1480-4-37
5. Library of Congress, L.: BIBFRAME Profile Editor (2018). http://bibframe.org/profile-edit/
6. Coyle, K.: RDF-AP, January 2017. https://github.com/kcoyle/RDF-AP, original-date: 2017–01-12T15:38:41Z
7. Enoksson, F.: DCMI: A MoinMoin Wiki Syntax for Description Set Profiles, October 2008. http://www.dublincore.org/specifications/dublin-core/dsp-wiki-syntax/
8. Hartig, O., Zhao, J.: Publishing and consuming provenance metadata on the web of linked data. In: McGuinness, D.L., Michaelis, J.R., Moreau, L. (eds.) IPAW 2010. LNCS, vol. 6378, pp. 78–90. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17819-1_10
9. Heery, R., Patel, M.: Application profiles: mixing and matching metadata schemas. Ariadne (25) (2000). http://www.ariadne.ac.uk/issue/25/app-profiles/
10. Hillmann, D.: Metadata standards and applications (2006). http://managemetadata.com/, publisher: Metadata Management Associates LLC
11. LD4P2: Sinopia Profile Editor (2019). https://profile-editor.sinopia.io/
12. Lebo, T., et al.: Prov-o: The prov ontology. W3C recommendation 30 (2013)
13. Li, C., Sugimoto, S.: Provenance description of metadata application profiles for long-term maintenance of metadata schemas. J. Documentation **74**(1), 36–61 (2018). https://doi.org/10.1108/JD-03-2017-0042
14. Malta, M.C., Baptista, A.A.: A method for the development of Dublin core application profiles (Me4dcap V0.2): detailed description. In: Proceedings of the International Conference on Dublin Core and Metadata Applications, p. 14 (2013)
15. Malta, M.C., Baptista, A.A.: A panoramic view on metadata application profiles of the last decade. Int. J. Metadata Semant. Ontol. **9**(1), 58 (2014). https://doi.org/10.1504/IJMSO.2014.059124
16. Moreau, L., et al.: The open provenance model core specification (v1.1). Futur. Gener. Comput. Syst. **27**(6), 743–756 (2011). https://doi.org/10.1016/j.future.2010.07.005
17. Nagamori, M., Kanzaki, M., Torigoshi, N., Sugimoto, S.: Meta-bridge: a development of metadata information infrastructure in Japan. In: Proceedings International Conference on Dublin Core and Metadata Applications, p. 6 (2011)
18. Nilsson, M., Baker, T., Johnston, P.: DCMI: The Singapore Framework for Dublin Core Application Profiles, January 2008. http://dublincore.org/specifications/dublin-core/singapore-framework/
19. Nottingham, M., Bryan, P.: JavaScript Object Notation (JSON) Patch, April 2013. https://tools.ietf.org/html/rfc6902

20. The DCAT Application profile for data portals in Europe (DCAT-AP), April 2019. https://github.com/SEMICeu/DCAT-AP, original-date: 2017–09-13T07:53:27Z
21. Svensson, L.A.R.V.: Negotiating Profiles in HTTP, March 2017. https://profilenegotiation.github.io/I-D-Accept-Schema/I-D-accept-schema
22. Svensson, L.G., Atkinson, R., Car, N.J.: Content Negotiation by Profile, April 2019. https://www.w3.org/TR/dx-prof-conneg/
23. Thalhath, N., Nagamori, M., Sakaguchi, T.: YAMA: Yet Another Metadata Application Profile (2019). https://purl.org/yama/spec/latest
24. Thalhath, N., Nagamori, M., Sakaguchi, T., Sugimoto, S.: Authoring formats and their extensibility for application profiles. In: Jatowt, A., Maeda, A., Syn, S.Y. (eds.) ICADL 2019. LNCS, vol. 11853, pp. 116–122. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34058-2_12
25. Thalhath, N., Nagamori, M., Sakaguchi, T., Sugimoto, S.: Yet another metadata application profile (YAMA): authoring, versioning and publishing of application profiles. In: International Conference on Dublin Core and Metadata Applications, pp. 114–125 (2019). https://dcpapers.dublincore.org/pubs/article/view/4055. ISSN 1939-1366