Towards describing version history of BCF data in the Semantic Web

Jyrki Oraskari[1], Oliver Schulz[1] and Jakob Beetz[1]

[1]Department of Design Computation, RWTH Aachen University, Aachen, Germany

Abstract

The buildingSMARTs BIM Collaboration Format (BCF) is a vendor-neutral standard to communicate BIM-based issues. The aspirations are well in line with open BIM and the Linked Building Data (LBD) community to use open data standards in Architecture, Engineering and Construction industry. Our previous research introduced the bcfOWL ontology to express BCF data using the open standards of the World Wide Web Consortium (W3C) as part of the LBD. The current server-based BCF-approach (BCF API) expresses changes in the model as event logs that are indirectly described. bcfOWL does not yet provide an equivalent implementation. This paper presents four different approaches for expressing these temporal changes as Linked Data. Two RDF-star approaches and a state construction inspired by the Ontology for Property Management (OPM) are introduced. Moreover, we show an event system method that is close to the original BCF API. To compare the approaches, queries to get the current data and search the history are provided on an external repository. Finally, the results and their adaptation are assessed.

Keywords

BCF, RDF-star, States, Version history, Linked Data

1. Introduction

The traceability of editions and design choices made over time is an integral part of the Architecture, Engineering and Construction (AEC) industry. Time after time, construction projects end up in court, where the responsibility for issues in the building or the planning process is litigated.

The BIM Collaboration Format (BCF) is mainly used in the AEC industry to communicate issues in the digital models. In the standard, an issue is described by a problem description (Topic) and an arbitrary number of Comments and virtual camera positions (Viewpoints) that locate the problem. It helps users assign responsibilities for issues and monitor their completion status. As buildingSMART stated in [1], BCF can be used to document quality assurance and quality checking for inter-domain coordination, annotating design alternatives, bidding items, expressing cost- or supplier information, reporting facility changes, and saving owner notes.

The bcfOWL ontology [2] was previously developed by the authors. It enables expressing BCF information in the Linked Building Data (LBD) context. The aspiration was to create a proposal

LDAC 2022: 10th Linked Data in Architecture and Construction Workshop, May 29, 2022, Hersonissos, Greece
Jyrki.Oraskari@dc.rwth-aachen.de (J. Oraskari); schulz@dc.rwth-aachen.de (O. Schulz); j.beetz@caad.arch.rwth-aachen.de (J. Beetz)
0000-0002-4723-3878 (J. Oraskari); 0000-0002-4722-4621 (O. Schulz); 0000-0002-9975-9206 (J. Beetz)

for an ontology for the core concepts of the BCF standard to represent the information in the Semantic Web using the best practices available. Thereby, bcfOWL is not intended to replace current integrations of the standard but rather to provide a data model that is self-documenting, queryable, machine-readable and extensible. With bcfOWL, it is still possible to have a web service that connects to the graph and provides the data via JSON responses.

In its server-based format (BCF API[1]), BCF comes with Event Services (REST API endpoints) that list changes made to the Topics and Comments. Since it was regarded as out of scope, bcfOWL does not yet provide an equivalent implementation.

In this paper, we study different approaches for including ways to express the history and evolution of issues in bcfOWL. The focal points are in expressing the source or provenance data, i.e. traceability of changes, changes made, the ability to reason behind the changes and enable queries on the history data. Although we concentrate on the BCF Topic - the core concept for describing issues in the standard - the study results should also apply to other BCF concepts. As non-repudiation is an essential aspect when describing the history of the data, it is one of this work's goals that the existing data content - once created - is not changed. Therefore, the constraint arises that the structure should only be based on addition, not removal operations.

This paper is organized as follows: Section 2 focuses on the current research around relating work. In Section 3, we introduce the different approaches for describing the history of BCF Topics and provide sample graphs. The following Section compares the approaches and provides measurements to evaluate their complexity. In Section 5 we discuss the results and the final Section concludes the work and gives an outlook for future work.

2. Related Work

2.1. BCF Events

In 2016, the BCF group discussed adding an array of history items to the BCF specifications [2]. They decided only to add an Event System to the BCF API, and since version 2.1, the API has defined four endpoints to query data modifications concerning Topics and Comments. The Topics are the central concept of an Issue, containing attributes - such as a status, an author, and a due date. The Comments are a messaging concept within the format used to communicate with other stakeholders in a BCF Project.

The overall structure of these endpoints services is mainly the same. With queries containing globally unique identifiers of Topics or Comments, the user can fetch all Events related to those identifiers. The returned Event contains the author, the date and which parameters were changed. The BCF API returns all its responses in a JSON serialization.

```
{"events": [{
        "type": "status_updated",
        "value": "Active"
       }]}
```

Listing 1: JSON snippet of the timed action in the Events System.

[1]BCF API: https://github.com/buildingSMART/BCF-API accessed 06.03.2022

[2]BCF history discussion: https://github.com/BuildingSMART/BCF-XML/issues/97 accessed 21.04.2022

The JSON body contains a list of timed actions, where every action represents an interaction with a Topic. Every action contains a list of Events that display which Topic attribute (the type) was changed to a new value (listing 1). The actions attributes are not sharing the types of the original Topics and Comments concepts but are represented in a combined string (e.g. *"status_updated"*). In total there are three different types of events: *created*, *updated*, and *deleted*.

The theory behind this data structure is that every time a Topic related data is created or edited, the server creates a log entry concerning the Topic change. Therefore, It is not enough to query only the Topic attributes to discover the complete picture of the history. All its Events have to be queried too.

2.2. Reification in RDF

Reifications are constructed by linking a node (the reification node) with the subject, predicate, and object of a statement in concern. The RDF reification mechanism, defined in the RDF Primer [3], is a known attempt to express statements about RDF assertions. Still, as Watkin and Nicole mentioned in [4], the reification is only indirectly related to the original statement. Instead of using reification, Watkin and Nicole suggested using Named Graphs introduced in [5]. It was also presented how secure digests (signed check-sums of the triple contents) can be used to affirm the author and integrity of the graph. Cassidy and Ballantine extended the use of reifications to express RDF Patches in the TriG syntax [6]. An RDF Patch is a group of reifications to describe changes in the model.

RDF-star [7] is an alternative, less wordy approach to describe reification in the Turtle syntax and SPARQL query language. It can be combined with standard reification and differs from the RDF pattern treating a triple as a subject. RDF-star introduces the concept of "Quoted Triples", which allows using an RDF Term (subject predicate object) as a subject or object of a triple.[3]

2.3. Describing States in Linked Building Data

Bonduel proposed a level system for the W3C Linked Building Data Community Group (W3C LBD CG)[4] consisting of three layers of complexity [8], whereas the third layer can describe states of things. The Ontology for Property Management (OPM) is also founded on this architecture and is based on a three-layer modelling. With OPM, changes of properties can be described with the help of states [9].

The Ontology for Managing Geometry (OMG) [10] builds on top of this level system so that either a thing can be:

- directly linked to the geometry description by a datatype or an object property (level 1).
- linked to one or many geometry nodes that are pointing to the description values (level 2), thereby adding the possibility to have many different geometric descriptions of the same thing.
- linked to a geometry node (as in level 2), but with an additional node in between. This allows the description of different states for the individual geometries (level 3).

[3] RDF-star and SPARQL-star https://w3c.github.io/rdf-star/cg-spec accessed 21.04.2022
[4] LBD Community Group: https://www.w3.org/community/lbd/ accessed 06.03.2022

2.4. Version Control and Versioning

Kiryakov and Ognyanov stated in [11] that a triple (or assertion) is the smallest changeable unit in an RDF graph. It implies that a change in a model can only be made by applying sets of triple removals and additions. However, in [12], *atomic graphs* - i.e. the surroundings of the blank notes - are the smallest unit to describe changes to avoid duplicating blank nodes. In this work, the context is how to express changes using bcfOWL, which is not relying on blank nodes. Therefore, no explicit blank nodes are needed.

In the light of the definition of Ball et al. [13], a version control system is a way to store and reconstruct past versions of data. Here we explore methods to express BCF data in RDF so that both aspects (store and reconstruct) are achievable. Other factors of version control, like the capability to undo and merge changes and synchronization, are out of this study's scope. On a graph level, versioning is also implemented by the different vendors. For example, GraphDB offers a "Data history and versioning" plugin [5] that can be queried for changes in the triples. However, this feature is not implemented in every graph storage and does not allow versioning on the RDF file level. Therefore, we decided to investigate how to express changes on the graph.

3. Approaches

The data models that we consider in this work are designed to reflect the design work responsibilities. Hence, each approach is crafted to remove no assertions from the model, but every modification adds new triples to the data model. We provide four different approaches for describing the data:

1. Event approach (Section 3.1)
2. States approach (Section 3.2)
3. Statements of Statements RDF-Star approach (Section 3.3.1)
4. Object Property Annotations RDF-Star approach (Section 3.3.2)

Further, a common scenario was chosen to ensure comparability between the variants. The use-case is described in Fig. 1.

We have chosen the example because it displays 1) how a single attribute that is always "updated" in the context of BCF (e.g. the status) can be handled and 2) how lists (the labels) are to be treated that can contain at every stage different values. These circumstances are common in issue management scenarios, where an issue can be marked as closed, but other participants disagree and reopen the issue. Although we focus exclusively on the Topic in this paper, the same types of changes also appear in the context of Comments. The third central concept of BCF - the Viewpoint - is not considered because it should never be changed according to the current definition of the buildingSMART standard. The alternative approaches are presented in the following subsections. The complete graphs of the approaches and example queries can be retrieved from our GitHub repository[6].

[5] GraphDB Documentation: https://graphdb.ontotext.com/documentation/9.8/standard/data-history-and-versioning.html accessed 05.03.2022

[6] GitHub repository with sample queries and graphs: https://github.com/Design-Computation-RWTH/LDAC2022_Dataset

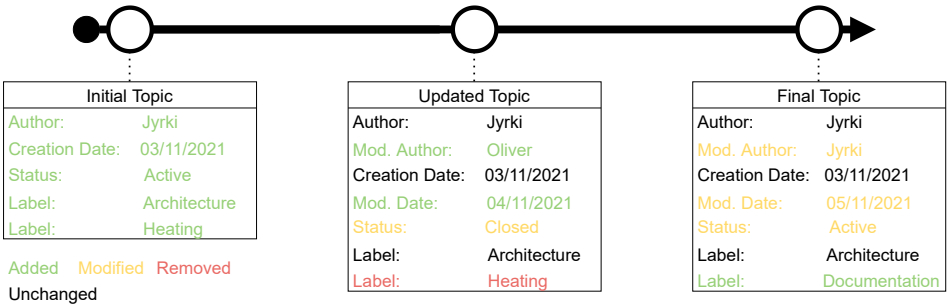

Figure 1: Jyrki creates a new Topic, which is the most current version of it now. The Topic contains parameters such as the creation date, a title, two labels (Architecture and Heating) and a status set to "Active". On the next day, Oliver sets the Topic's status to "Closed" and removes the label "Heating". Another day later, the Topic gets reopened (set to "Active") by Jyrki and the label "Documentation" gets added.

3.1. Event approach

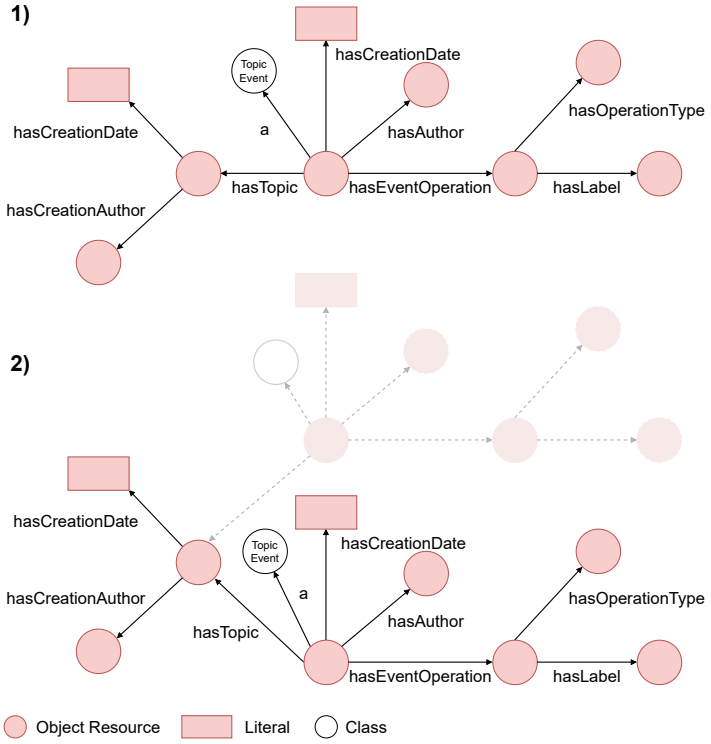

Figure 2: The Event System approach is based on the BCF API. A Topic Event points to the related Topic and its operations that specifically define what type of interactions have happened. For example, if a property was removed, updated or created. Next to the type of operation, the attribute's value is stated.

In the BCF API, the Topic Events Service is a log of changes for the Topic. It lists creations, updates, and removals of specific Topic event types. In the Events approach, we describe this logbook-like behaviour by creating a new log entry with all operations created as soon as an interaction with the Topic happens.

In the BCF APIs Event Service, the Events coexists next to the Topics and Comments. This behaviour mainly comes from the fact that the BCF standard does not describe how the data is saved but is a way of communicating it. With bcfOWL, we have, on the one hand, a data format (RDF) and, on the other hand, the SPARQL query language for communicating the information from and to the graph. This is where the structure of the bcfOWL Events differs from the BCF API by not creating "redundant" data and allow using the Events to infer the Topics information.

An Event is described by the class of "bcfOWL:TopicsEvent" that states its author, the creation date and which Topic it belongs to. Furthermore, the object-type property "bcfOWL:hasEvent-Operation" links the Event to the different operations that occur in it (see Fig. 2). These operations state if it is an update, a creation or a deletion, and state the property's current value. So if the status of a Topic was set from "Active" to "Closed", this would be described as an "Update" operation, and the current value would be "Active". The event approach comes with a change to the overall structure of bcfOWL. The Topics are now not responsible for providing all the attributes. Everything that can be changed in the context of BCF has now to be inferred from the Events. The newest value for an attribute is also its current value. One exception has to be made here because the property "bcfOWL:hasLabel" usually provides a list. Therefore, it is not enough to select the newest Event, but the query has to consider if a label was removed at one point in time or not.

3.2. States approach

The "States" approach is splitting the attributes of the "bcfOWL:Topic" class into two parts and is transferring all the attributes that can change throughout the project to "bcfOWL:TopicState". This approach is similar to OPM and OMG (Section 2.3), but due to the structure of the Topics, it requires only two levels. Every State provides the whole picture of its current status, meaning it provides all attributes, even if just one value of a new state has changed. A State comes with a creation author and a creation date. These are equivalent to the "Modified Author" and "Modified Date" of the BCF API. When retrieving the current values of a Topic, the State with the newest date has to be queried. (see Fig. 3)

3.3. RDF-star approaches

RDF-star can be used to make statements about triples and thereby attach meta information to them (Section 2.2). The following two approaches use RDF-star to express all interaction- and change-related content. To describe a change, we need to state when something was changed, who authored it, and semantics about whether it was an addition or removal. Since RDF-star allows many possible approaches, we decided to provide two different examples, 1) by using statements of statements and 2) by using object properties linking to annotation triples.

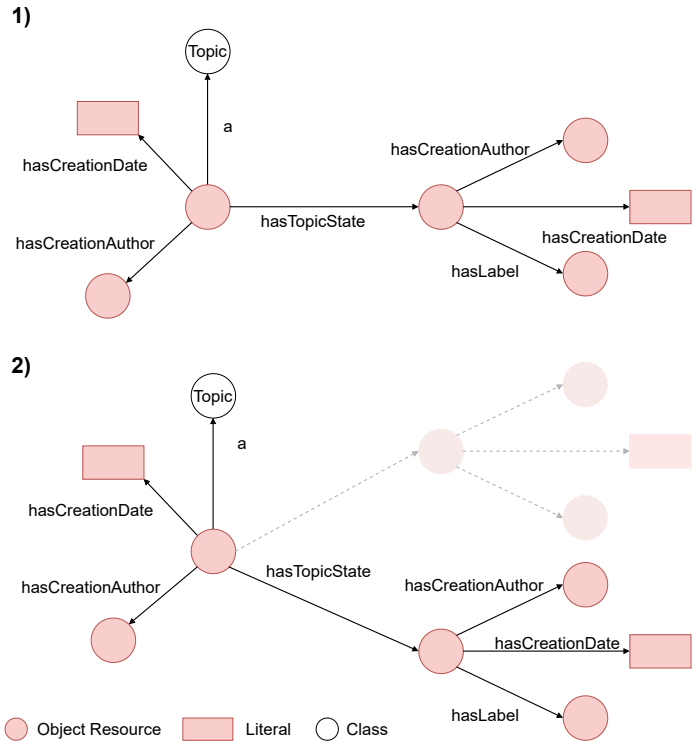

Figure 3: The different states are referenced by the Topic with the property “bcfOWL:hasTopicState”. The states provide the complete picture of the Topic during a specific time. In order to find the current version of a Topic we have to query for the newest date. An example query is provided on the GitHub repository.

3.3.1. Statements of Statements RDF-star approach

By using RDF-star with statements of statements (shown in Fig. 4) we can describe with the first statement when e.g. the triple for a "bcfOWL:hasTopicStatus" was added by using "bcfOWL:wasAdded" and a time string. This statement then gets annotated with its author. Since attributes can change many times - a status can change from "Active" to "Closed" and back again to "Active" - these annotations are constantly added as soon as a triple is interacted with. The annotation with the newest date should always be considered the Topic’s current state. Same as in the previous sections, the bcfOWL:hasLabel is an exception here, and all the labels should be treated as current as long as they were not annotated as removed in their current state.

3.3.2. Object Property Annotations RDF-star approach

The other RDF-star approach adds only one level of statements. There, every change-related property of the Topic is annotated by the „bcfOWL:change“ (Fig. 5) object property, which points to a sub-graph that states the author, the date and a state (added or removed).

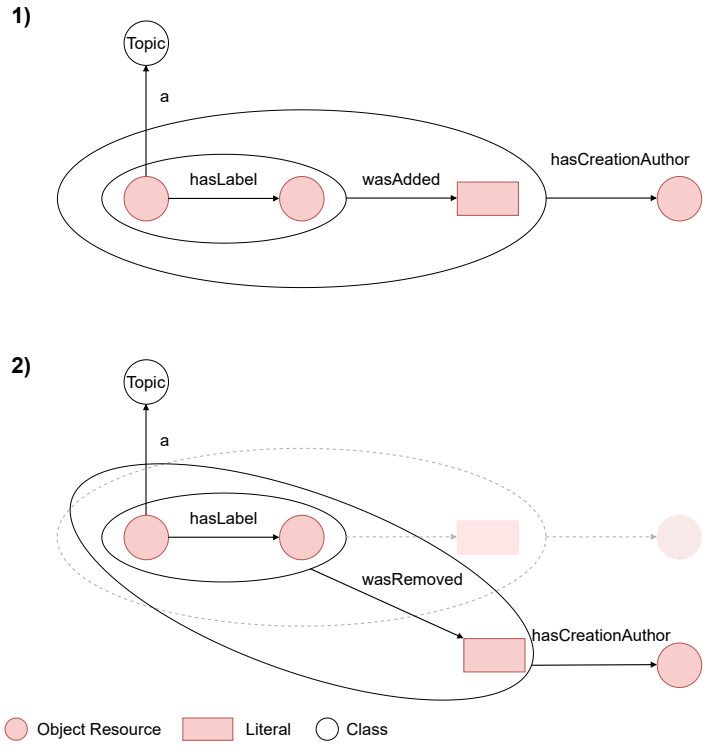

Figure 4: Statements can be marked as “added” or “removed” with a time-stamp. The author of this change is added as a statement of the first statement. Even though the word itself states a “deletion”, resources are never removed from the graph to allow retrieval of the complete history of the Topics.

4. Results

The evaluation criteria were chosen to attempt to measure the expected simplicity of the various integration approaches. Therefore, we measured the triple counts, the maximum reasoning steps needed, diameters of the models, and the number of variables needed to query the models using SPARQL. We expect that shorter RDF path lengths imply simplicity for writing queries. The same applies for more concise graphs. A lower number of triples in the graphs and the variables for the queries were considered "good".

All four handcrafted alternative data sets express the same information. The initial triple counts of the data and the added triples that describe the changes were counted using the Apache Jena API. They are shown in Table 1 and the counted triples are listed in separate files at the GitHub repository (Section 3).

The States approach had the lowest initial triple count and had the second-lowest number of added triples. The Statements of Statements approach performed best regarding the final triple count.

Although reification is not present in the triple counts of RDF-star, the reification triples are implicitly present in the SPARQL-star functions: SUBJECT, PREDICATE, OBJECT. Therefore, it

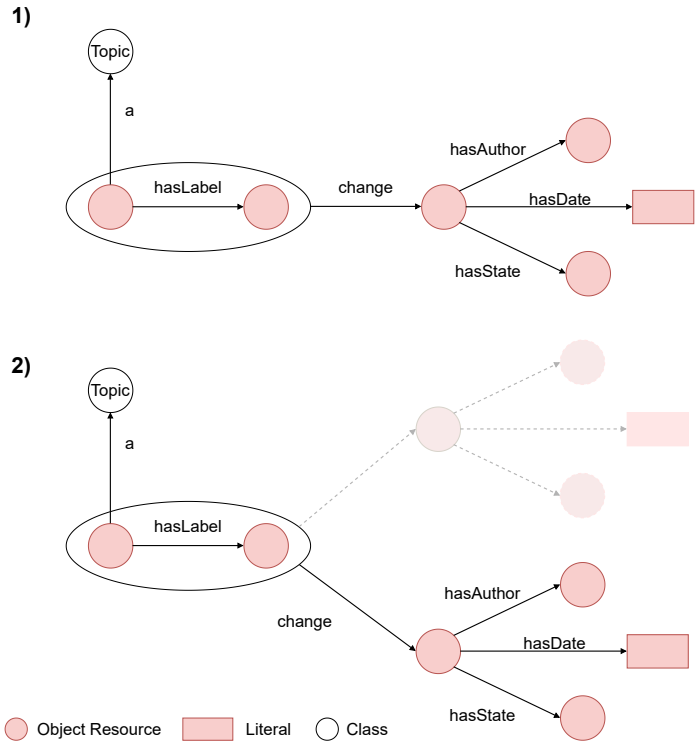

Figure 5: Object property annotations only contain one level of statements and point to an object that contains all change-related information.

is possible to make the approaches comparable by making the relations explicit in the graph. So, instead of many connected sub-graphs in the Apache Jena data presentation, as shown in Table 1, the alternative models have only a single connected sub-graph. It means they are traversable from any node in the model to any node as an undirected graph. Thus, it allows having an idea of the complexity of the graph. For example, the diameter of the model and the maximum steps from a Topic to a value can be evaluated and compared.

Three SPARQL queries were written for each alternative approach to estimating how easy the model is to query. The queries were to fetch 1) the current Topic data, 2) a state at a single point of time, and 3) the initial values for the Topic data. The needed numbers of SPARQL variables were counted (see Table 1). The Event system needed more variables to handle the two different object type cases: direct literal values and a list of enumerated type objects. The complete data is shown in the repository.

5. Discussion

Overall, the results were well in line with our experience when handcrafting the different approaches.

The Event approach (Section 3.1) is closely tailored to mirror the structure of the BCF API

Table 1
Comparison of the approaches.

Approach	Events	States	RDF-Star: Statements of Statements	RDF-Star: Object Property Annotations
Initial triples in the data model	29	14	21	24
Triples added in the first update	12	8	7	13
Triples added in the final update	15	9	7	17
Triples at the end	57	31	35	54
Max. steps to a leaf.	3	2	3	3
Initial graph: model diameter	4	3	4	4
Final graph: model diameter	6	4	5	6
Final graph: The number of connected sub-graphs in Jena Model	1	1	23	8
SPARQL variables needed to query:				
- the current state	9	6	6	6
- state at a specific time	9	6	6	6
- state at initial time	6	3	4	5
blue - good, red comparatively bad				

data model. So, the graph is never depicting the current state directly since it has to be inferred out of the different Event Operations, making the graph harder to query.

The State approach (Section 3.2) is related to current developments in the LBD group and is following their examples. We would argue that it is easy to implement in bcfOWL and ontologies in general. Moreover, as its name says, it always expresses the complete data snapshot of a Topic, even though sometimes just a single value has changed. It is relatively space-saving compared to the other methods (Table 1), although there is a risk of storing redundant parameters in the graph due to the complete state description. It lacks the granularity of the other approaches since it is impossible to see how a single parameter has changed from one state to the next without calculating the differences between the data sets.

However, with RDF-star, it is possible to describe states and track granular changes. Thus, in a way, it represents a middle ground between the other two methods. Both variants, which are based on RDF-star, are non-invasive to the original structures of the ABox statements. A legacy bcfOWL graph could thus easily be annotated, which could then describe the history of the triples. The RDF-star approach is not exclusive to BCF but could be a valuable addition in general. Because there is no deletion in our model, there will not be any dangling nodes in the system. While we cannot enforce non-deletion, each quoted triple in RDF-star is a separate assertion and thus, should not be considered dangled even if the referred node would be deleted.

Since timestamps are used in the Statements of Statements approach (Section 3.3.1), it has the limitation that the annotations could not be distinguished in the hypothetical case when an exactly simultaneous annotation of a triple would be made. This limitation does not exist with the Object Property Annotation approach (Section 3.3.2).

The RDF-star approach's drawback is that it is not yet supported by all the graph database providers and Linked Data tools. If the scope of a project depending on RDF-star is limited to a graph database supporting this feature, no issues should be expected. Nevertheless, suppose the goal is to federate the data over multiple and distributed graphs where there is no control over the chosen database type. In that case, it can lead to compatibility issues. Furthermore, it is not

yet clear where statements made about statements can end. Without a clear structure, these statements can be chained endlessly, adding complexity to the graphs.

6. Conclusion and Future Work

In this work, we have shown four different approaches to describe the history of changes in BCF Topics. All four versions display the same information, only using different structures.

As the change information is an integral part of the issue management, our focus has been on how to make the version data available as a semantic web. So, instead of addressing the change management at a higher level, e.g. the named graph level or trying to use GIT, we have focused on the RDF graph level. We hypothesise that the approach allows better to trace editions and design choices made over time to the data.

We have pointed out that information should never be removed from the graph or changed, although enforcing this requirement may be challenging. Hence, in future work, we want to study the options for signing changes in the graph to assure the stakeholders that the values have not been altered. It should also be evaluated how the different approaches would perform in distributed and container-based environments. Also, one engaging direction is to study how provenance data can be expressed and used in the BCF context. The provenance indicated in the graphs in this paper is currently only a placeholder, as the provenance of the data in bcfOWL has not yet been explicitly considered. Here the PROV ontology [14] is an interesting candidate. It can be studied how to use the ontology to express the ownership of the data, who played a role or contributed creating it, and how the content has been changed.

RDF-star performed reasonably well in our studies, so we think it should be further investigated for use in the LBD domain. Nevertheless, although RDF-star can be easily integrated into existing graph structures, we have not yet found examples of how this approach could be described in a Tbox statement for reasoning. An ontology extension for bcfOWL using RDF-star could help us tackle the complexity of statements made about statements.

The results of the study serve as a basis for further development for the bcfOWL ontology. For example, the shortest graph diameter and the shortest maximum path from a Topic to a value node can imply if the model is a good candidate as data representation for the BCF change data. In order to verify the results, we further have to simulate these approaches using larger data sets, covering a more considerable amount of changes and history data.

7. Acknowledgments

The EU had funded this research through the H2020 project BIM4REN.

References

[1] buildingSMART International, Ltd., Bim collaboration format (bcf) - an introduction, 2022. URL: https://technical.buildingsmart.org/standards/bcf/.

[2] O. Schulz, J. Oraskari, J. Beetz, bcfowl: A bim collaboration ontology, 2021. URL: https://www.cibw78-ldac-2021.lu/, international Workshop on Linked Data in Architecture and Construction, LDAC ; Conference date: 11-10-2021 Through 15-10-2021.
[3] F. Manola, E. Miller, B. McBride, RDF Primer (2004). URL: https://www.w3.org/TR/rdf-primer/.
[4] E. R. Watkins, D. A. Nicole, Named graphs as a mechanism for reasoning about provenance, in: Asia-Pacific Web Conference, Springer, 2006, pp. 943–948.
[5] J. J. Carroll, C. Bizer, P. Hayes, P. Stickler, Named graphs, provenance and trust, in: Proceedings of the 14th international conference on World Wide Web, 2005, pp. 613–622.
[6] C. Bizer, R. Cyganiak, The trig syntax. 2007, URL http://sites. wiwiss. fu-berlin. de/suhl/bizer/TriG/Spec/TriG-20070730 (2007).
[7] O. Hartig, Foundations of rdf* and sparql*:(an alternative approach to statement-level metadata in rdf), in: AMW 2017 11th Alberto Mendelzon International Workshop on Foundations of Data Management and the Web, Montevideo, Uruguay, June 7-9, 2017., volume 1912, Juan Reutter, Divesh Srivastava, 2017.
[8] B. Mathias, Towards a PROPS ontology, 2018. URL: https://github.com/w3c-lbd-cg/lbd/blob/a76232b682a979ff707ad0150c9336ff5b7b0f8e/presentations/props/presentation_LBDcall_20180312_final.pdf.
[9] M. Holten Rasmussen, M. Lefrançois, M. Bonduel, C. Anker Hviid, J. Karlshøj, Opm: An ontology for describing properties that evolve over time, in: CEUR Workshop Proceedings, volume 2159, CEUR Workshop Proceedings, 2018, pp. 24–33.
[10] A. Wagner, M. Bonduel, P. Pauwels, R. Uwe, Relating geometry descriptions to its derivatives on the web, in: Proceedings of the 2019 European Conference for Computing in Construction, European Council on Computing in Construction (EC3), 2019, pp. 304–313. URL: http://hdl.handle.net/1854/LU-8633667. doi:10.35490/ec3.2019.146, iSSN: 2684-1150.
[11] D. Ognyanov, A. Kiryakov, Tracking changes in rdf (s) repositories, in: Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, 2002, pp. 373–378.
[12] S. Auer, H. Herre, A versioning and evolution framework for rdf knowledge bases, in: International Andrei Ershov Memorial Conference on Perspectives of System Informatics, Springer, 2006, pp. 55–69.
[13] T. Ball, J.-M. Kim, A. A. Porter, H. P. Siy, If your version control system could talk, in: ICSE Workshop on Process Modelling and Empirical Studies of Software Engineering, volume 11, Citeseer, 1997.
[14] P. Missier, K. Belhajjame, J. Cheney, The w3c prov family of specifications for modelling provenance metadata, in: Proceedings of the 16th International Conference on Extending Database Technology, 2013, pp. 773–776.