PATH-TRACING SEMANTIC NETWORKS TO INTERPRET CHANGES IN SEMANTIC 3D CITY MODELS

S. H. Nguyen*, T. H. Kolbe

Geoinformatics, Department of Aerospace and Geodesy, TUM School of Engineering and Design,
Technical University of Munich (TUM), Germany - (son.nguyen, thomas.kolbe)@tum.de

Commission IV, WG IV/9

KEY WORDS: Change Detection, Change Interpretation, Urban Digital Twins, Semantic 3D City Models, CityGML, Graphs.

ABSTRACT:

Recent years have seen a significant and rapid rise in the development and deployment of many urban digital twins worldwide. Urban digital twins provide a central platform for incorporating data and knowledge from different sources and fields, and can thus be used for many urban-related processes such as urban planning, analysis, monitoring and visualization. One of the key requirements of digital twins in general and urban digital twins in particular is the continuous, bidirectional data flow between the physical entity and its digital counterpart. Consequently, this requires the ability to both detect and understand changes between different temporal versions of the physical and digital entities. In the context of smart cities and semantic 3D city models however, only a few studies have addressed this so far. This is due to the facts that: (1) Semantic 3D city models (mostly encoded in CityGML) contain multifaceted information modelled in a complex inheritance hierarchy, which complicates their change detection process; (2) As cities constantly evolve over time, matching their datasets often results in a large number of changes, which must be further analysed and processed to produce meaningful information; (3) Individual changes in the datasets are multi-layered as well as often correlated and cannot be fully understood without considering their context in the city model; and (4) Different types of changes are perceived and interpreted differently by different stakeholders, meaning that the outcome of the change detection and interpretation process must ultimately serve the human factor. Therefore, to address these challenges, this research proposes a Path-tracing Semantic Network (PSN) to interpret detected changes in semantic 3D city models. The framework represents the multidimensional nature of changes together with stakeholders in a semantic network, where their interrelations can be analysed explicitly using graph-based path-tracing methods.

1. INTRODUCTION

An increasing number of urban digital twins are being developed and deployed by many cities, districts and companies worldwide. An urban digital twin can be thought of as a digital representation of a physical city, including its assets, processes and services. Urban digital twins provide a central platform for incorporating information from heterogeneous sources, including (real-time) Internet of Things (IoT) sensor readings, remote sensing data, etc. Urban digital twins can therefore be deployed for a wide range of applications, such as urban planning, monitoring and visualization, allowing a risk-free virtual environment to simulate, experiment and evaluate future policies. Alone the cost benefits of using urban digital twins for efficient urban planning are expected to reach USD 280 billion by 2030, according to (ABI Research, 2021).

Despite the still many definitions to date across various application domains, the general consensus remains that one of the key requirements of digital twins in general and urban digital twins in particular is the continuous, bidirectional feedback between the physical entity and its virtual counterpart (Fuller et al., 2020). This synchronization of the different states of the physical and digital entity is often called the twinning process, which consists of two complementary halves as illustrated in Figure 1: (1) **Physical-Digital Twinning**: Changes that occurred in the real world (indicated by the blue arrow between *Physical* and its changed state *Physical′*), such as by physical behaviours or processes, are reflected to the digital entity (indicated by the blue arrow between *Digital* and the final state *Digital′*); and (2) **Digital-Physical Twinning**: Changes that occurred in the digital entity (indicated by the red arrow between *Digital* and *Digital′*), such as products of simulation or planning process, are also reflected back to the physical world (indicated by the red arrow between *Physical* and the final state *Physical′*).

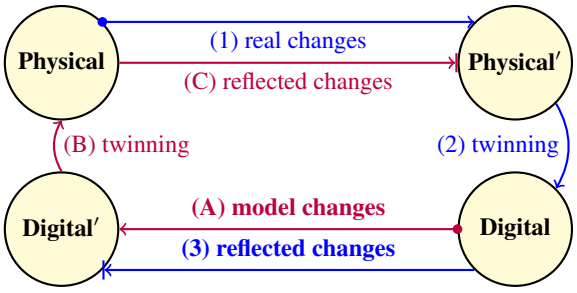


Figure 1. Data flow of changes between a physical entity and its digital counterpart in a digital twin.

Without identifying changes in step *(1)* and *(A)* shown in Figure 1, the twinning process would have to rebuild or replace the physical and digital entity regularly respectively. In the context of smart cities and urban digital twins, where semantic 3D city models serve as the digital representation of cities, the effects of a lack of change detection capability become apparent in use cases, where older city models are being replaced entirely by newer ones. This not only wastes time and computational resources but also removes meaningful information

* Corresponding author

that can be used to analyse cities' evolution over time. This is due to the fact that changes cannot be simply considered as pure deviations and differences in the datasets, they each also have a meaning, a context as well as complex and implicit interrelations with each other. Thus, acquiring the ability to both detect and understand changes between different temporal versions of a city model will not only enable efficient identification of changes in the digital datasets, but also provide useful insights into the real changes that occurred in the physical world, which is also an important application of urban digital twins themselves. Thus, this research focuses on the steps *(A)* and *(3)* shown in Figure 1, namely to help interpret changes of semantic 3D city models employed as digital representations of cities.

Although change detection is not new, only a few studies have discussed this so far in the context of semantic 3D city models (mostly stored in CityGML, an open-standard data model for storage and exchange of digital 3D city models and landscapes). Earlier studies include (Cobena et al., 2002), (Wang et al., 2003) and (Redweik and Becker, 2015) and were used to compare the XML tree representations of city models. However, due to the graph nature of CityGML, other studies such as (Falkowski and Ebert, 2009) and (Agoub et al., 2016) showed the potential and expressiveness of graphs as a representation of CityGML documents. (Nguyen et al., 2017) then proposed a new approach to comparing arbitrarily large CityGML datasets using a graph database. On the other hand, (Nguyen and Kolbe, 2021) explained the importance and advantage of not only identifying but also understanding and evaluating changes with respect to different groups of stakeholders. In general, the major problems of solving change detection and interpretation in semantic 3D city models can be summarized as follows:

1. CityGML stores information from different aspects (namely semantic, geometry, topology and appearance) together in one place using a highly structured inheritance hierarchy and a number of syntactic rules, which allows flexible ways to define city objects, but at the same time complicates the change detection process;

2. Comparing two temporal versions of the city model of the same city often results in a large number of changes, which may quickly become unmanageable and incomprehensible from a human perspective, especially if many of them are located at the dataset level only and do not reflect any real changes in the physical world;

3. Most studies consider changes as isolated deviations in the data, they contain however (often hidden) multifaceted information and complex interrelations among each other, and thus cannot be fully understood without also considering their semantic context in the city model;

4. The human factor plays the biggest role in interpreting changes, since different stakeholders have different expectations and interests in different types of changes, which are difficult to provide explicit quantifiable measures due to their multidimensionality, high implicitness and variability over time.

Thus, to overcome the above-mentioned challenges, this research proposes a framework called **Path-tracing Semantic Network (PSN)** for interpreting changes in semantic 3D city models. A multi-layered semantic network is used to model and describe changes, stakeholders and their interrelations. A graph-based path-tracing method is introduced to help analyse and evaluate such relations. Section 2 provides an overview of some studies related to this research. Section 3 explains the concepts and structure of the proposed PSN, including its two key components, namely the semantic network and the path-tracing process. The framework's application in interpreting changes of semantic 3D city models is also explained. Section 4 concludes and summarizes this research.

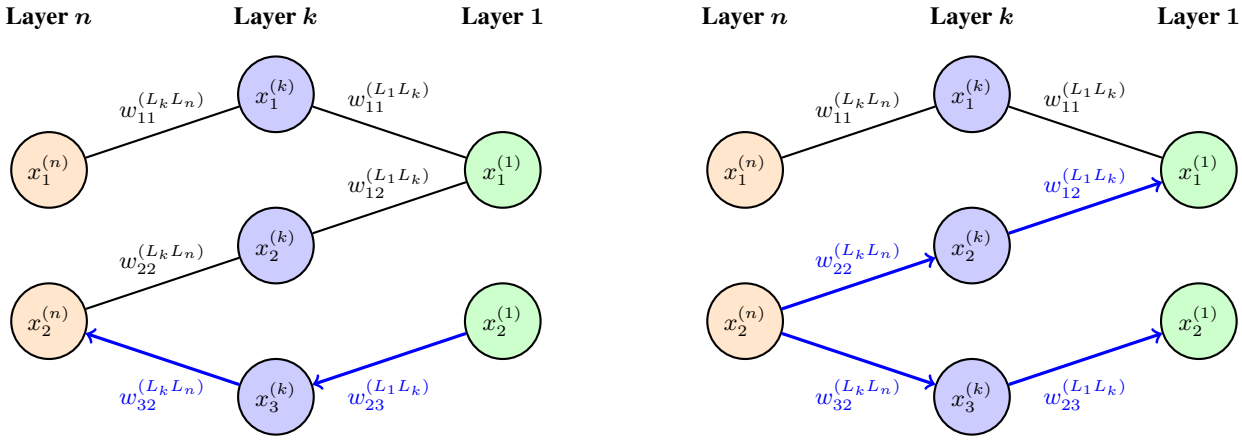
2. RELATED WORK

Change detection or data comparison algorithms (whose applications are often called *diff* tools) such as (Wagner and Fischer, 1974), (Hunt and Szymanski, 1977) and (Myers, 1986) are well-established for unstructured plain text files. Despite often also being available as text files, most semantic 3D city models are however stored in CityGML (Gröger et al., 2012, Kolbe et al., 2021), which is a Geography Markup Language (GML) application schema, which, in turn, is a grammar of the Extensible Markup Language (XML). Therefore, comparison algorithms and methods designed specifically for more complex exchange data formats such as XML are required. Since XML documents are hierarchically structured and can thus be considered as trees, (Cobena et al., 2002) presented a fast and memory-efficient *diff* algorithm for XML data by matching unchanged XML subtrees between the older and newer version. (Wang et al., 2003) proposed *X-Diff*, an effective algorithm that considers key XML characteristics using standard tree-to-tree correction techniques. The authors argued that, despite being substantially more difficult to execute compared to ordered tree model, matching XML documents using their unordered tree model generates more accurate results.

Utilizing the unordered tree model proposed by (Wang et al., 2003), (Redweik and Becker, 2015) later extended their work to better include some spatial and geometric information available in the tree representation of CityGML documents. However, their approach did not include the use of the XML Linking Language (XLink), which is often utilized in CityGML to allow the explicit reusing or linking of previously defined elements in the same document, since including XLinks would form (undirected) cycles in the tree representation of CityGML documents, which would contradict the definition of trees themselves. This means that, despite being XML-based, CityGML documents are basically considered as graphs and cannot be limited to trees only (Schade and Cox, 2010).

Early graph adaptations and representations of CityGML include (Falkowski and Ebert, 2009) that presented a graph-based schema for storing, analysing and managing city objects. The authors employed a *TGraph*, which is a directed graph, where vertices and edges are typed, ordered and enriched with attributes. The model schema integrated parts of the many information aspects of CityGML, namely geometry, topology, semantic and appearance. Later, (Agoub et al., 2016) focused on the potential of storing graph representations of complex data models with well-defined objects, attributes and relations like CityGML in a database. The authors discussed the difficulties of storing and managing such complex objects using Relational Database Management Systems (RDBMS) and provided a lightweight method for mapping and storing objects of various Open Geospatial Consortium (OGC) standards (including CityGML) into the graph databases Neo4j and ArangoDB. On the other hand, (Yao, 2020) discussed utilizing the expressiveness, flexibility and extendability of graphs

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-X-4-W2-2022-217-2022 | © Author(s) 2022. CC BY 4.0 License.



(a) Forward path tracing starting from Layer 1

(b) Backward path tracing starting from Layer n

Figure 2. An example of a bidirectional Path-tracing Semantic Network (PSN), where $x_i^{(k)}$ indicates the weight of vertex $v_i^{(k)}$ of layer L_k and $w_{ij}^{(L_{k_1}L_{k_2})}$ indicates the weight of edge connecting vertices $v_i^{(k_1)}$ and $v_j^{(k_2)}$ of layer L_{k_1} and L_{k_2} respectively.

to enhance the interoperability of CityGML Application Domain Extensions (ADEs) with existing spatial relational databases, such as the 3D City Database (3DCityDB) (Yao et al., 2018). The authors defined a set of graph transformation rules, which can be dynamically applied to the graph representations of the XML Schema Definition (XSD) files of CityGML ADEs. The transformed graphs can then be converted to relational database schemas in SQL. The above-mentioned studies showcased the potential and versatility of the graph representations of CityGML, its underlying data model and its related data in many different applications; they however did not explain how CityGML documents can be matched using graphs.

Exact graph matching, or the graph isomorphism problem, is the well-known computational problem of determining whether two finite graphs G and H are isomorphic, meaning whether there exists a bijective function $f : V(G) \rightarrow V(H)$ between the vertex sets of G and H, such that for every pair of adjacent vertices u and v of G, the resulting $f(u)$ and $f(v)$ of H are also adjacent. This has been addressed in many prominent studies such as (Babai, 2015). Graph isomorphism is commonly employed in computer vision, pattern recognition, etc., where the number of vertices of both graphs are often kept the same. Due to the nature of cities' evolution however, change detection between graph representations of city models aims not at finding an exact match of the dataset, but rather at detecting where changes did occur, since the number of vertices of such graph representations are often not the same. Inexact graph matching, or homomorphic graph matching, where f is injective instead of bijective, searches for a match of a smaller graph within the bigger one (Bengoetxea, 2002). This is equivalent to the sub-graph matching problem or the sub-graph isomorphism problem.

(Nguyen et al., 2017) introduced the use of graphs to store, manage and compare CityGML documents in the graph database Neo4j. The authors employed an R-Tree on top of building footprints, which can be utilized to find suitable sub-graphs that are prime candidates for the sub-graph isomorphism problem. The research was one of the first to provide an actual working implementation of the mapping and matching process of arbitrarily large CityGML datasets using a graph database. However, the study did not provide further instructions on how to interpret the detected changes, especially in large numbers. (Nguyen and Kolbe, 2020) presented a multi-perspective approach to understanding such changes with respect to different groups of stakeholders by dividing changes into different categories. The relevance relations between changes and stakeholders can then be expressed using a table. (Nguyen and Kolbe, 2021) proposed a refined model of changes as well as stakeholders in the context of urban digital twins with semantic 3D city models as one of their key components. The research further provided two mathematical ways to flexibly model the varying relevance relations between different groups of stakeholders and different types of changes, namely using the relevance matrix as well as the relevance graph. The above-mentioned studies served as one of the first steps towards achieving automatic and comprehensive change detection and interpretation. They did not however further address the fact that changes are also multifaceted, often correlated and cannot be fully understood without considering their context in the city model. Moreover, further instructions and mathematical methods are required to explicitly model and describe the implicit interrelations between not only changes and stakeholders, but also among changes and stakeholders themselves.

Therefore, to fill in these gaps, this research proposes a Path-tracing Semantic Network (PSN) to provide a better and more comprehensive view of changes, stakeholders as well as their complex interrelations.

3. PATH-TRACING SEMANTIC NETWORK (PSN)

3.1 Definition

A Path-tracing Semantic Network (PSN) is a graph $G = (V, E)$ consists of a set V of vertices and a set E of edges. The vertex set V is comprised of partitions L_k called layers. At least two layers are required in a PSN, namely an input and output layer. Layers are serially connected, meaning a layer L_k can only have incoming connections from its previous layer L_{k-1} and outgoing connections to its subsequent layer L_{k+1}. The input layer L_1 does not have incoming edges and the output layer L_n does not have outgoing edges. Each vertex or node $v_i^{(k)}$ of layer L_k is assigned a weight $x_i^{(k)} \in \mathbb{R}$. An edge $e_{ij}^{(L_kL_l)}$ is a directed or an undirected connection from vertex $v_i^{(k)}$ to vertex $v_j^{(l)}$ and is assigned a weight $w_{ij}^{(L_kL_l)} \in \mathbb{R}$. Figure 2 illustrates an example of such network.

A PSN is structurally similar to common Artificial Neural Networks (ANNs). This is an intended design choice to facilitate the usage of PSN in future deep learning applications. However, compared to common neural networks, a Path-tracing Semantic Network is additionally characterized as follows:

1. A **multi-layered semantic network** with well-defined rules for vertices and edges is used to model and capture interrelations between concepts, where a vertex additionally represents a concept or an object, while a connection between two vertices has a semantic meaning, typically representing the real-world or modelled relationships between its incident vertices (as explained in Section 3.2);

2. Thus, a vertex $v_i^{(k)}$ neither requires incoming edges from all input vertices in the previous layer L_{k-1}, nor outgoing edges to all output vertices in the subsequent layer L_{k+1}. A connection is set only if it represents a meaningful relation between vertices;

3. A PSN employs **graph-based path-tracing techniques** globally on the entire network to analyse the semantic meaning of its components (as explained in Section 3.3). In order to facilitate forward and backward path tracing between the input and output layer, a connection between two vertices must be bidirectionally traversable (i.e. by using an undirected edge or two directed opposite edges);

4. In a PSN, a layer can be partitioned further into a set of serial or parallel sub-layers to better model and process semantic interrelations between concepts and objects.

Therefore, PSNs were designed to deal with models of data rich in semantic information and complex interrelations. The use of a connected multi-layered network provides not only an explicit representation of often implicit interactions between objects, but also an expressive and intuitive way to describe and capture the complexity nature of objects and their relations.

3.2 Semantic Network for Changes and Stakeholders

A semantic network is the first key component of the framework Path-tracing Semantic Network (PSN). In this section, one such semantic network shall be used to represent changes, stakeholders and their complex relations on multiple levels in an example of a building, whose roofs have been vertically raised by an offset while keeping their original shape (see Figures 3 to 5).

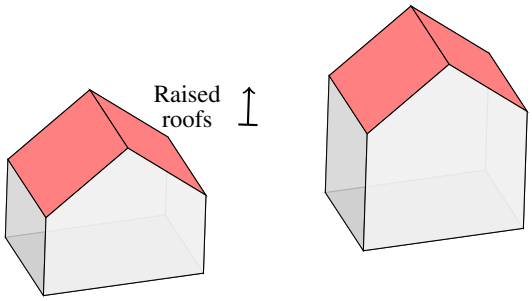


Figure 3. An example of a building before (left) and after (right) its roof surfaces have been raised without changing their form.

The semantic network comprises of five different layers, namely the input Concept Layer L_1, the Change Type Layer L_2, the Reasoning Layer L_3, the Actor Role Layer L_4 and the output Stakeholder Layer L_5, which shall be explained in the following sections. In this network, all vertices and edges are weighted with real numbers. These values represent the interest and relevance levels between adjacent vertices. For instance, in a directed network, a weight $w_{ij}^{(L_k L_l)}$ denotes the relevance value of vertex $v_i^{(k)}$ of layer L_k towards vertex $v_j^{(l)}$ of layer L_l, while in an undirected network, this weight applies for both vertices in each direction. If $w_{ij}^{(L_k L_l)} = 0$, the edge $e_{ij}^{(L_k L_l)}$ can be omitted, meaning there exists no relation between vertices $v_i^{(k)}$ and $v_j^{(l)}$. In the network shown in Figure 5, edges are undirected. The directed red and blue edges indicate the direction of the path-tracing process, which shall be explained in Section 3.3. In case of different weights in each direction, it is recommended to use directed edges instead. Alternatively, positive and negative weights can also be assigned to undirected edges to appoint a consistent path-tracing direction over the entire network. Infinite values can be used to explicitly prioritize or bypass specific vertices and edges while traversing.

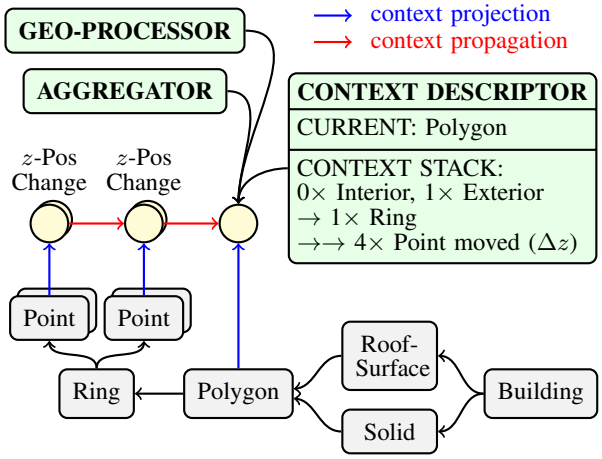


Figure 4. The three components Context Descriptor, Geo-processor and Aggregator for context extraction.

3.2.1 Context Layer (L_1): The Context Layer L_1 is the input layer of the semantic network. It combines changes and their context together in a graph structure. Since a change is not simply an individual isolated deviation in the data, context is needed to provide useful insights into the changes' origin, scope and their relations with other changes. In semantic 3D city models, the context containing semantic and spatial information of a change can be extracted from its parent and ancestor elements stored in the dataset. Depending on the methods used to represent city models, the context extraction process can be implemented and applied correspondingly. For instance, by using graphs suggested by (Agoub et al., 2016) and (Nguyen et al., 2017), the context of CityGML objects can be derived directly from their graph representation. The context extraction process consists of three main components: the **context descriptor**, the **geo-processor** and the **aggregator**. These are illustrated in Figure 4 on the basis of a simplified graph representation of a roof surface that has been raised vertically as shown in Figure 3.

Given a directed graph representation of a city model with a single source vertex (i.e. a vertex with only outgoing edges), the context extraction process starts with all sink vertices (i.e. vertices with only incoming edges) at the bottom and iterates upward for each parent of the current vertex until the source is reached. In each iteration, the process searches for indicators of a detected change at the current location. If this is the case, the context of the current vertex is projected and collected by

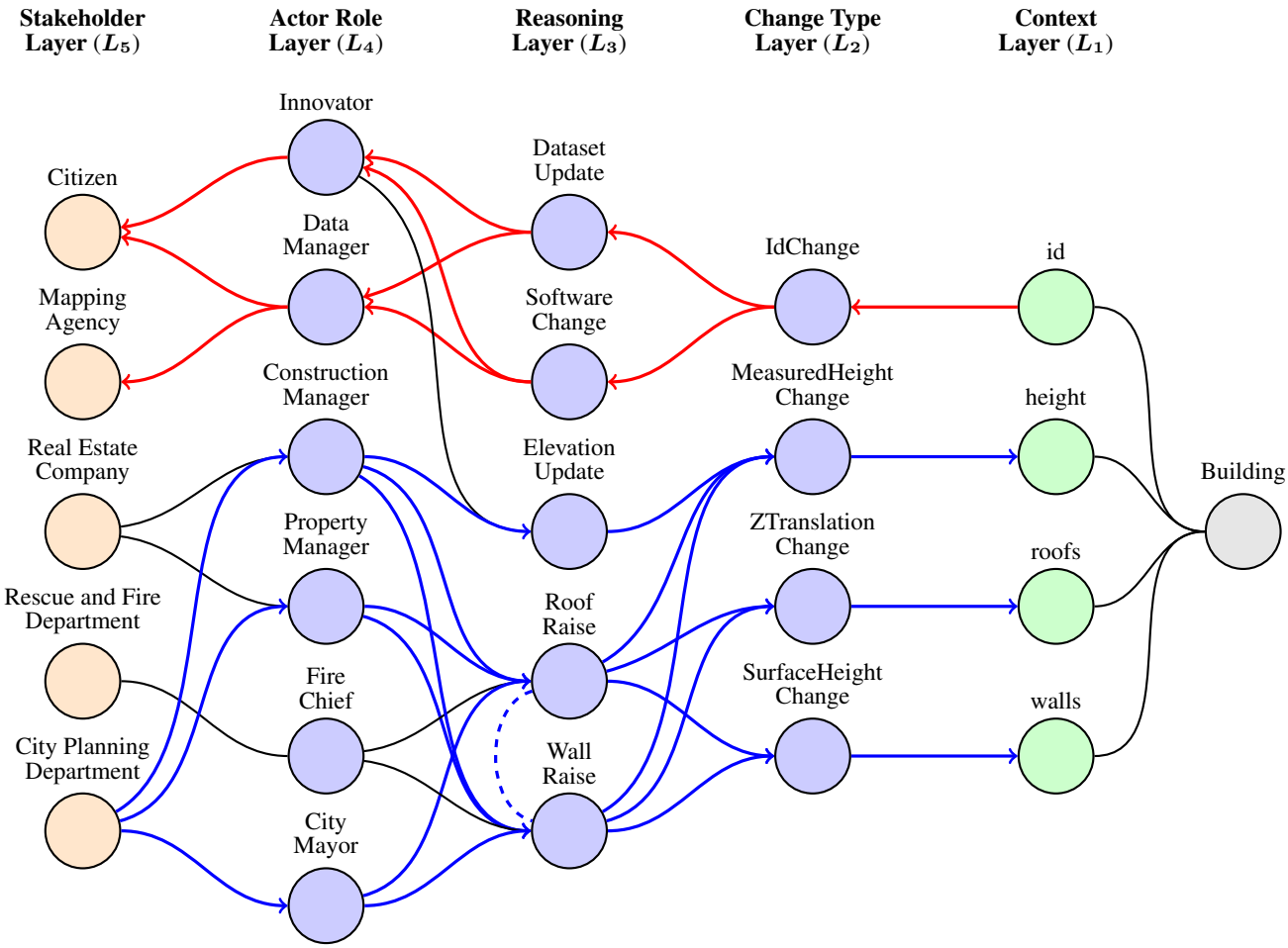


Figure 5. An example of a two-way Path-tracing Semantic Network (PSN) used to model and analyse changes, stakeholders and their relations. A forward and backward path-tracing are shown in red and blue respectively.

the context descriptor. The context descriptor may store such context information in a linked list (such as a queue or a stack) to retain the order of its recorded contents. Depending on the type of the current element, a geo-processor with its own well-defined rules is applied to process the geometric information collected by the context descriptor so far. Based on the results, the aggregator determines whether the detected changes are semantically and geometrically related. For instance, as shown in Figure 4, based on the context collected so far from a polygon object (i.e. composed of one exterior linear ring with four control points and no interiors) stored in the context descriptor, the geo-processor is able to determine that all four control points of the polygon were translated by the same Δz along the Oz axis. This indicates that the entire roof surface was moved vertically by Δz without changing its form. The aggregator therefore considers the changes in the four control points as one single translation change of the roof surface. Moreover, the aggregator also regulates the context descriptor and the geo-processor accordingly when a convergence or divergence in the directed graph is encountered, such as between *Polygon*, *RoofSurface* and *Solid*, as well as *RoofSurface*, *Solid* and *Building*. These patterns are used to group semantically, geometrically and topologically related changes in a bottom-up approach, thus condensing the number of detected changes significantly while revealing their hidden correlations at the same time. The vertices *id*, *height*, *roofs* and *walls* shown in the Context Layer L_1 in Figure 5 represent the aggregated results of the context extraction process.

3.2.2 Change Type Layer (L_2): The context information extracted and aggregated in previous layer L_1 as explained in Section 3.2.1 can be forwarded to the Change Type Layer L_2 to further classify changes, provide scope and therefore potentially identify local, clustered or global (systematic) changes in the dataset. Conceptual models of changes that are compatible with semantic 3D city models, such as proposed by (Nguyen and Kolbe, 2021), can be used to populate the vertices in this layer. The framework proposed in this research allows flexible structuring of each layer, meaning that entire layers can be reorganized into a corresponding graph representation of a class inheritance hierarchy as long as they satisfy the requirements of the network given in Section 3.1. The more fine-grained the conceptual modelling is, the more types of changes this layer can capture and classify. Graph representation of unassigned (abstract) superclasses are considered local only to the current layer and shall not be further forwarded to the subsequent layers. This can be achieved by marking or "colouring" all such vertices and their corresponding edges as local to be filtered out by the path-tracing process at a later stage. Alternatively, (signed) infinite weights can be assigned to such vertices and edges to achieve the same effects.

3.2.3 Reasoning Layer (L_3): Real-world objects are often interconnected by various logical and physical processes. A change to an object may also cause changes to others, or a single process can trigger changes to a number of connected objects. In semantic 3D city models, the semantic, geometric

and topological connections of city objects can be expressed explicitly using e.g. CityGML. These information aspects are however often lost or hidden after the change detection process is complete. Thus, in this section, an additional layer called the Reasoning Layer L_3 is introduced to help capture and model the implicit correlation and causal effects of changes of objects.

Each vertex of the Reasoning Layer represents an action that caused detected changes in the datasets. Based on their effects, these actions can either be classified as changes of the physical reality or modifications to the digital representations, as shown in steps *(1)* and *(A)* of Figure 1. Similarly to the types of changes in the Change Type Layer L_2, a conceptual model of such actions can be provided for layer L_3. Then, the correlation and causal relations between changes in layer L_2 and real-world actions in layer L_3 can be expressed explicitly by weighted edges in the semantic network. For instance, as shown in Figure 5, in the direction from layer L_2 to layer L_3, a change in the measured height of a building can be caused by an update in the elevation of the entire city model, or raised roofs, or raised walls. These actions are thus called correlated due to the changed measured height being their common causal effect. Vice versa, in the direction from layer L_3 to layer L_2, an action to raise the roofs of a building (i.e. to lift the existing roof surfaces up vertically without changing their form) may cause a change in the measured height, a vertical change in the position of roof surfaces as well as changed vertical size of wall surfaces. Therefore, these types of changes are called correlated due to the action to raise the roofs being their common cause.

Formally, when traversing the network in a specific direction, if outgoing edges from at least two vertices $v_i^{(k-1)}$ and $v_j^{(k-1)}$ from layer L_{k-1} converge into one single vertex $v_p^{(k)}$ of layer L_k, then there exists a correlation relation between vertices $v_i^{(k-1)}$ and $v_j^{(k-1)}$, and $v_p^{(k)}$ is called their correlation effect. Depending on the semantic meaning of layers L_{k-1} and L_k, a causal relation may additionally be implied, in which case the vertex $v_p^{(k)}$ is called either a common cause or a common causal effect of both $v_i^{(k-1)}$ and $v_j^{(k-1)}$ depending on the interpretation direction. This process can thus be applied in both traversing directions if the network is bidirectional. Furthermore, if there exists another vertex $v_q^{(k)}$ of layer L_k, to which outgoing edges of the same vertices $v_i^{(k-1)}$ and $v_j^{(k-1)}$ of layer L_{k-1} converge, then both $v_p^{(k)}$ and $v_q^{(k)}$ are also considered correlated. Depending on the semantic meaning of such vertices, their relation may be further refined as a causation. For example, both actions to raise the roofs and walls of a building in layer L_3 have connections to the same types of changes in layer L_2. This indicates a correlation relation between the two actions (denoted by a dashed connection in Figure 5). Analysing their semantic role further shows that the action to raise roofs causes the action to raise all walls in a one-to-many causal relation. The conceptual model of actions used in layer L_3 can therefore be additionally extended with such correlation relations between actions.

Moreover, the correlation and causal relations are not only limited to two neighbouring layers, the analysis can also be applied to a number of serially connected layers. Note that the traversing direction must be consistent across layers during the process. The meaning of such correlation and causal relations is interpreted based on the semantic meaning and models of the traversed layers. If the correlation analysis is applied to the entire network (i.e. including both the input and output layer), then it is called the path-tracing process, which shall be explained in Section 3.3.

3.2.4 Actor Role Layer (L_4) and Stakeholder Layer (L_5):

Changes in layer L_2 are caused by actions in layer L_3, which are in turn issued by humans. This section shows how the human factor can be represented as layers in the semantic network. Past studies have shown the complexity of the relations between stakeholders and different types of changes in semantic 3D city models. On one hand, like changes, the interest levels vary over time. On the other hand, stakeholders may perceive different types of changes differently depending on their current professions, positions and roles in the process. Thus, to allow meaningful analysis and interpretation of changes with respect to stakeholders, two layers are proposed to represent stakeholders, namely the Actor Role Layer L_4 and the output Stakeholder Layer L_5 as shown in Figure 5.

By decoupling roles from stakeholders as a separate layer, the network is able to build direct connections between actions in layer L_3 and their actors in layer L_4. Roles are also a more expressive and robust way to describe the functional positions of a stakeholder in the process, since a stakeholder can have many different roles at the same time. The Stakeholder Layer L_5 is the last and output layer of the network. It represents stakeholders based on their physical characteristics, e.g. as individual beings (citizens), companies (private sectors) and organizations (non-governmental and governmental organizations), etc. Similarly to previous layers, a conceptual model for actors in layer L_4 and stakeholders in layer L_5, such as proposed by (Nguyen and Kolbe, 2021), can be employed to populate the network.

3.3 Graph-based Path-tracing Analysis

A Path-tracing Semantic Network (PSN) comprises of two key components: a **multi-layered semantic network** (as described in Section 3.2) and a **path-tracing process**, the latter of which shall be discussed in this section. The name "path tracing" of this framework is inspired by the technique path tracing in the field of computer graphics, which simulates realistic global illumination of a 3D scene by starting at the individual pixels on the objects' surface and following along the many light rays bounced between objects until the light source is reached (Kajiya, 1986). Light paths that did not reach the light source shall be discarded. Thus, using similar terminology, the one-way path tracing process of the framework PSN starts with the input layer L_1, then follows the many graph paths between layers until the output layer L_n is reached. Paths that did not reach the target layer shall be discarded. The two-way version of the path-tracing process can additionally start with the output layer L_n first and arrive at the input layer L_1. Path tracing in the direction from the input layer L_1 to output layer L_n is called *forward path tracing*, while the version in the opposite direction is called *backward path tracing*. These are illustrated in Figures 2a and 2b as well as in red and blue paths in Figure 5 respectively.

To ensure that the path-tracing process terminates, the network must either be a directed acyclic graph in each tracing direction, or it can contain cycles that are only traversable for a finite number of times. For bidirectional path tracing, the network must either be undirected or traversable in both directions. In the former case, edge weights are the same for both directions, while in the latter case, different weights for each direction can be assigned to the same connection between two vertices.

As one of its applications, the PSN can be employed to model, analyse and interpret changes of semantic 3D city models with respect to stakeholders. In contrast to local or clustered

processes such as the correlation analysis introduced in Section 3.2.3, the path-tracing process takes place on top of the entire network. Based on the example of raised roofs of a building shown in Figure 3 and the corresponding semantic network illustrated in Figure 5, the path-tracing process is applied in the following analyses:

1. **Changes-Stakeholders Analysis:** This process determines how relevant a change is to different actions, actors and ultimately stakeholders. Forward path tracing is employed in this case. For instance, as shown in the red paths in Figure 5, a detected change in the identifier of a building may indicate an update in the dataset or a software change. Consequently, this finding may be of interest to innovators and data managers, who are represented as citizens and mapping agencies. Thus, the results of this analysis are useful when a number of changes have been detected and need to be evaluated with respect to stakeholders;

2. **Stakeholders-Changes Analysis:** This process determines which roles, actions and ultimately types of changes are relevant to a specific stakeholder. Backward path tracing is employed in this case. For instance, as shown in the blue paths in Figure 5, a city planning department may have a role in urban construction, property management and policy making. They may therefore be interested in the changes indicating the raised roofs and walls of a building. Thus, the results of this analysis are required to provide stakeholders a better and human-readable overview of only interesting changes among many less relevant others stored in a database.

Therefore, by employing forward and backward path tracing in the semantic network, meaningful in-depth information and insights of the often hidden relations between concepts and objects can be achieved. This is done by evaluating the accumulated weights of all traced paths within the network in a specific direction. The accumulated weight w_P of a traced path $P = \left(v_{i_1}^{(1)}, v_{i_2}^{(2)}, \ldots, v_{i_n}^{(n)}\right)$ can be defined for forward path tracing as follows:

$$w_P = \sum_{k=1}^{n-1} x_{i_k}^{(k)} w_{i_k i_{k+1}}^{(L_k L_{k+1})} \tag{1}$$

where $v_{i_k}^{(k)}$ is the i_k-th vertex of layer L_k with weight $x_{i_k}^{(k)}$, $1 \leq i_k \leq |L_k|$ with $|L_k|$ as the number of employed vertices of layer L_k, and $w_{i_k i_{k+1}}^{(L_k L_{k+1})}$ is the weight of the edge connecting $v_{i_k}^{(k)}$ with $v_{i_{k+1}}^{(k+1)}$. Note that the vertex weight $x_{i_n}^{(n)}$ of the target layer L_n is excluded from w_P. This can be included if needed.

The path P has a maximum length of $n-1$ edges, since the network has n layers (excluding auxiliary vertices used locally within layers to represent unused classes, etc. of the conceptual models). There exist at most $\prod_{k=1}^{n} |L_k|$ such paths. If the length of P is not exactly $n-1$, then this path is not fully traced, meaning the target layer is not reachable. Otherwise, among fully traced paths, a minimum or maximum value of their accumulated weights (depending on the use cases and weight values) can be calculated to determine the most fitting candidate for further analyses. This corresponds to the shortest or longest path problem respectively. However, it is often the case where not only one but multiple candidate paths must be considered. As a result, paths weighted below a certain threshold can be ignored. The remaining candidate paths can then be sorted for further analyses based on their weights. The value of the above-mentioned threshold is determined depending on the specific use cases. For example, for the Changes-Stakeholders as well as Stakeholders-Changes analysis mentioned previously, weights of vertices and edges are given as real (normalized) numbers to represent the relevance values between changes and stakeholders. The most fitting candidates are therefore the fully traced paths that are weighted above a defined threshold ideally close to the maximum value of all accumulated weights.

Given a directed network, since no edges between vertices within the same layer exist (or can be neglected during the path-tracing process), the adjacency properties of two layers L_k and L_{k+1} can be described using the following modified submatrix A_k of their adjacency matrix:

$$A_k = \left(x_i^{(k)} w_{ij}^{(L_k L_{k+1})}\right) \in \mathbb{R}^{|L_k| \times |L_{k+1}|} \tag{2}$$

The adjacency matrix A of the entire network thus becomes:

$$A = \begin{array}{c} \\ \boldsymbol{L_1} \\ \boldsymbol{L_2} \\ \vdots \\ \boldsymbol{L_{n-1}} \\ \boldsymbol{L_n} \end{array} \begin{array}{c} \begin{array}{ccccc} \boldsymbol{L_1} & \boldsymbol{L_2} & \boldsymbol{L_3} & \ldots & \boldsymbol{L_n} \end{array} \\ \begin{pmatrix} 0 & \boldsymbol{A_1} & 0 & \ldots & 0 \\ 0 & 0 & \boldsymbol{A_2} & \ldots & 0 \\ & & & \ddots & \\ 0 & 0 & 0 & \ldots & \boldsymbol{A_{n-1}} \\ 0 & 0 & 0 & \ldots & 0 \end{pmatrix} \end{array} \tag{3}$$

Then, the value $a_{ij}^k \in A \circ_f A \circ_f \ldots \circ_f A = A^k$ corresponds to the accumulated weights of a path of length k between vertices v_i and v_j of the acyclic directed network, where $\circ_f$ is a modified matrix multiplication such that for each matrix $P = (p_{ij}) \in \mathbb{R}^{m \times l}$ and $Q = (q_{ij}) \in \mathbb{R}^{l \times s}$, $P \circ_f Q = (r_{ij}) \in \mathbb{R}^{m \times s}$, with $r_{ij} = \max\left(f\left(p_{it}, q_{tj}\right)\right) \forall t \in [1, l]$. Depending on the use cases, a minimum function may be preferred. The function f is defined for each real value of x and y so that $f(x, y) = 0$ if $xy = 0$, or $x + y$ otherwise. This ensures the weights of each path can be accumulated correctly. Since layers are serially connected, only the "diagonal" $(A_1 A_2 \ldots A_{n-k})$ of A^k is populated with non-zero values. As k increases, the diagonal moves upwards. Hence, all fully traced paths of length $n-1$ can be found in the submatrix located in the top right corner confined by the first $|L_1|$ rows and last $|L_n|$ columns of A^{n-1}. This can be utilized to reduce the size and computation of A^k.

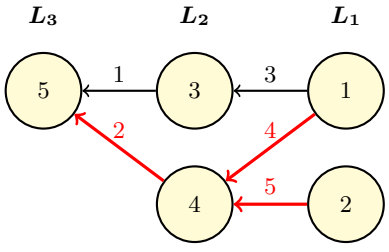


Figure 6. An example of a weighted and directed network.

For instance, the network illustrated in Figure 6 has the following adjacency matrix according to Equation (3):

$$A^2 = \left(\begin{array}{cc|ccc} 0 & 0 & 3 & 4 & 0 \\ 0 & 0 & 0 & 10 & 0 \\ \hline 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 8 \\ \hline 0 & 0 & 0 & 0 & 0 \end{array}\right)^2 = \left(\begin{array}{cc|ccc} 0 & 0 & 0 & 0 & \mathbf{12} \\ 0 & 0 & 0 & 0 & \mathbf{18} \\ \hline 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \end{array}\right)$$

4. CONCLUSION AND FUTURE WORK

This study proposes a Path-tracing Semantic Network (PSN), a flexible framework for modelling, analysing and interpreting changes of semantic 3D city models. A PSN consists of two key components: a semantic network and a path-tracing process on top. The semantic network is organized as layers to better capture and represent the multidimensional nature of changes and their relations not only to stakeholders but also among themselves. Based on the vertex and edge weights given in the network, the path-tracing process is employed to evaluate the modelled layers and find changes that are most relevant to a specific stakeholder or vice versa, find stakeholders that are most likely interested in a specific type of change.

This framework provides both an intuitive and explicit way to capture and describe complex interrelations between concepts and objects that are often hidden or implicit and thus difficult to observe and model. Due to its graph nature, a PSN can be quickly realized and coupled with existing data in many graph databases. Another advantage of a PSN is its scalability, interoperability, full extendability and customization thanks to its modular design. Additionally, the semantic network can be applied not only to express correlations between objects, but also to provide a correlation check on their correctness, or to predict correlated behaviours (such as a raised roof should also predict a change in the measured height of a building and vice versa). A PSN is compatible with common Artificial Neural Networks (ANNs) and could be extended for future use cases.

In the near future, the proposed framework Path-tracing Semantic Network (PSN) shall be applied and evaluated using real-world scenarios, where a semantic network shall be developed to model, identify and interpret most common types of changes with respect to stakeholders in semantic 3D city models. Furthermore, the framework could be employed to contribute to the ongoing efforts of realizing the automatic data flow required in urban digital twins in the context of smart cities and semantic 3D city models. However, a PSN can also be applied in other application areas and use cases, where information is multifaceted, often implicit, variable and highly interconnected.

REFERENCES

ABI Research, 2021. Urban Planning and Digital Twins. Technical Report AN-5416, Allied Business Intelligence, Inc.

Agoub, A., Kunde, F., Kada, M., 2016. Potential of Graph Databases in Representing and Enriching Standardized Geodata. *Dreiländertagung der DGPF, der OVG und der SGPF*, 36.

Babai, L., 2015. Graph Isomorphism in Quasipolynomial Time. *arXiv e-prints*, arXiv:1512.03547.

Bengoetxea, E., 2002. Inexact Graph Matching using Estimation of Distribution Algorithms. PhD thesis, Citeseer.

Cobena, G., Abiteboul, S., Marian, A., 2002. Detecting Changes in XML Documents. *Proceedings 18th International Conference on Data Engineering*, IEEE, 41–52.

Falkowski, K., Ebert, J., 2009. Graph-based Urban Object Model Processing. *City Models, Roads and Traffic (CMRT'09): Object Extraction for 3D City Models, Road Databases and Traffic Monitoring-Concepts, Algorithms and Evaluation, Paris, France*, 9.

Fuller, A., Fan, Z., Day, C., Barlow, C., 2020. Digital Twin: Enabling Technologies, Challenges and Open Research. *IEEE Access*, 8, 108952-108971.

Gröger, G., Kolbe, T. H., Nagel, C., Häfele, K.-H., 2012. OpenGIS(R) City Geography Markup Language (CityGML) Encoding Standard, Version 2.0. OGC.

Hunt, J. W., Szymanski, T. G., 1977. A Fast Algorithm for Computing Longest Common Subsequences. *Commun. ACM*, 20(5), 350–353.

Kajiya, J. T., 1986. The Rendering Equation. *SIGGRAPH Comput. Graph.*, 20(4), 143–150.

Kolbe, T. H., Kutzner, T., Smyth, C. S., Nagel, C., Roensdorf, C., Heazel, C., 2021. *OGC City Geography Markup Language (CityGML) Version 3.0 Part 1: Conceptual Model Standard*. Open Geospatial Consortium. International Standard.

Myers, E. W., 1986. An O(ND) Difference Algorithm and Its Variations. *Algorithmica*, 1(1-4), 251–266.

Nguyen, S. H., Kolbe, T. H., 2020. A Multi-Perspective Approach to Interpreting Spatio-Semantic Changes of Large 3D City Models in CityGML Using a Graph Database. *Proceedings of the 15th International 3D GeoInfo Conference 2020*, ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, VI-4-W1-2020, ISPRS, London, 143–150.

Nguyen, S. H., Kolbe, T. H., 2021. Modelling Changes, Stakeholders and their Relations in Semantic 3D City Models. ISPRS (ed.), *Proceedings of the 16th International 3D GeoInfo Conference 2021*, ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, VIII-4/W2-2021, New York University, ISPRS, 137–144.

Nguyen, S. H., Yao, Z., Kolbe, T. H., 2017. Spatio-semantic Comparison of Large 3D City Models in CityGML Using a Graph Database. *Proceedings of the 12th International 3D GeoInfo Conference 2017*, ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, IV-4/W5, ISPRS, Melbourne, 99–106.

Redweik, R., Becker, T., 2015. Change Detection in CityGML Documents. *3D Geoinformation Science: The Selected Papers of the 3D GeoInfo 2014*, Springer, 107–121.

Schade, S., Cox, S., 2010. Linked Data in SDI or How GML is not about Trees. *Proceedings of the 13th AGILE International Conference on Geographic Information Science-Geospatial Thinking*, 1–10.

Wagner, R. A., Fischer, M. J., 1974. The String-to-String Correction Problem. *J. ACM*, 21(1), 168–173.

Wang, Y., DeWitt, D. J., Cai, J. Y., 2003. X-Diff: An Effective Change Detection Algorithm for XML Documents. *Proceedings of the 19th International Conference on Data Engineering 2003*, 519–530.

Yao, Z., 2020. Domain Extendable 3D City Models – Management, Visualization, and Interaction. Dissertation, Technical University of Munich, Munich.

Yao, Z., Nagel, C., Kunde, F., Hudra, G., Willkomm, P., Donaubauer, A., Adolphi, T., Kolbe, T. H., 2018. 3DCityDB - a 3D Geodatabase Solution for the Management, Analysis, and Visualization of Semantic 3D City Models Based on CityGML. *Open Geospatial Data, Software and Standards*, 3(5), 1-26.