Representation of Concurrent Points of View of Urban Changes for City Models

John Samuel Samuel, Sylvie Servigne, Gilles Gesquière

This is a post-peer-review, pre-copyedit version of an article published in Journal of Geographical Systems. The final authenticated version is available online: at: https://doi.org/10.1007/s10109-020-00319-1

Representation of Concurrent Points of View of Urban Changes for City Models

John Samuel · Sylvie Servigne · Gilles Gesquière

the date of receipt and acceptance should be inserted later

Abstract Cities evolve over time and their evolution is often studied using material objects and historical documents. Based on available evidence, researchers propose various hypotheses explaining the probable evolution, both imaginary and factual. Furthermore, city models are used to model and visualize 3D structures and semantic information of the cities. With the help of versioning of city objects, it is possible to represent temporal changes of city structures. And with the support for representing scenarios, it is possible to represent different possible sequences of urban changes. In this article, we formalize a set of rules for representation of concurrent points of view of researchers related to urban changes based on standard city model. The goal is to understand how the cities have evolved, what were the key changes and allow exchange between various hypotheses (or processes). We developed a proof-of-concept named $UrbanCo^2Fab$ based on these rules to demonstrate the highly potential use cases of our proposition.

1 Introduction

Cities are under constant evolution. In the study of understanding historical past of a city, it is important to understand not only the changes that occurred but also how these changes occurred. Historical documents like old photographs, paintings, postal cards etc. give a snapshot [Peuquet and Wentz, 1994] or version [Van Ruymbeke et al., 2015] of the city at a particular instant or a period of time. Take for instance, a council meeting report describing the discussion on demolition of a building and an associated photograph that proves its actual demolition at a later period. But historical documents are not just confined to such factual elements of the urban past. They may also give an imagined view of the city by different actors. Though such imagined views like project plans may have never been implemented, their mere propositions in some cases may have left a lasting impact on the cities.

To understand and study these aspects, researchers are increasingly looking for solutions that can be used to model structural and semantic information concerning various city objects at different points of time. Geographical information systems (GIS), especially maps are now commonly used for navigational purposes, currently serving the main purpose of getting the most recent version of the city. But historians and urban planners require options to navigate

Univ. Lyon, CPE Lyon, LIRIS, F-69616, Lyon, France · Univ. Lyon, INSA Lyon, LIRIS, F-69622, Lyon, France · Univ. Lyon, Université Lumière Lyon 2, LIRIS, F-69622, Lyon, France E-mail: john.samuel@liris.cnrs.fr · sylvie.servigne@liris.cnrs.fr · gilles.gesquiere@liris.cnrs.fr

through the historical timeline to comprehend several aspects of the urban fabric. They want to spatially and temporally annotate their document corpus in GIS along with several other domain-specific information before finally being able to pose complex queries and navigate or explore urban data through data visualization approach. In other words, a support for nD (or multidimensional) solutions that can represent and visualize the 3D (spatial), temporal as well as semantic information is required.

Collaboration among multiple actors is important in every scientific domain. Researchers and experts [Van Ruymbeke et al., 2017, Van Ruymbeke et al., 2018] propose, discuss and debate various hypotheses on probable urban evolution. Different actors base their decisions on facts available to them at a given moment of time. It is important to save these valuable discussions and the resulting hypotheses to get an overview of the various points of view concerning the study of urban changes. Projects proposed or imagined by different urban planners may or may not come to completion. Nevertheless, these projects give multiple alternate scenarios envisioned by the proposers in contrast to the existing reality. Thus we also need to be able to represent different proposed scenarios. Lessons from the past failed projects also serve as a guiding factor for the future projects of urban planners. This work here is focused on the past. Urban planning will require changes that are not provided here in order to have a focus on the urban changes in the past. Nevertheless, our work may be seen as a common ground for urban planning.

Finally, considering the ever-changing software landscape, one commonly discussed problem is the incompatability of past softwares in newer platforms. Hence there is an increasing demand for data-driven solutions based on standards (e.g standard city models) in order to promote data interoperability and data sharing.

Considering all these requirements, we introduce and formalize a new method to represent concurrent points of view in a 4D context in this article. Our proposed method lets multiple researchers propose and share various scenarios of the city evolution. Section 2 describes in detail the study of urban evolution and its various requirements. We take a look at several related works in section 3. Section 4 details versions and workspaces and also presents their formalization. We present $UrbanCo^2Fab$ in section 5 and briefly describe the proof of concept. Section 6 finally concludes the article and presents the future course of actions.

2 Study of Urban Evolution

As cities expand both vertically and horizontally, the study of urban evolution has taken prime importance in several research circles especially among the urban planners. As cities expand both vertically and horizontally, the study of urban evolution has taken prime importance in several research circles especially among the urban planners and historians. Lessons from the past serve as a guidance for the future project plans of a city. Urban evolution study involves several domains: geography, social science, urban planning, history etc. In our work, we are interested in representing this nD (or multimensional) urban evolution information: 3D structural, temporal and several domain-specific information. Precisely, our document corpus consists of old postal cards, aerial views, photographs, project plans (both successful and failed), municipal meeting notes. This corpus is currently used by us for selective case studies for the manual recreation of a building or a city sector at different points of time using standard city models.

Cities have their stories [Fields, 2012] to tell, right from the first human establishment to its current state. Even though it seems utopian to have a complete and continuous description of the urban changes, historical documents and anecdotes may be considered as witnesses or footprints of changes. Researchers study this urban evolution at various scales. While some of them focus on the changes of buildings [Stefani et al., 2010] of historical importance, others take it to much

larger scale of city sectors [Simon, 2012] or to the entire scale of the city [Renolen, 2000, Lefebvre et al., 2008, Rollier-Hanselmann et al., 2014, Billen et al., 2012] or to even larger space [Harbelot et al., 2013, Chaturvedi et al., 2017] in a much more generic approach. They focus on changes in different features of city objects. Highly granular changes like on those on the level of buildings [Stefani et al., 2010] take into consideration various possible transformations like annexation, union etc. of buildings, whereas those on a coarser level consider only the features of city objects like roads, buildings, bridges etc. without going into details.

City object features are used to describe 3D structural, temporal, semantic and thematic information concerning the objects. For any given city object, the feature values change during any given period of time. These could be structural changes, like the number of storeys of a building or they could be related to its use (e.g., administrative, religious use of a building, use of a bridge for vehicles, pedestrians etc.). It is interesting to observe the evolution of these changes in feature values for given city objects for the study of urban evolution. Such studies are usually based on available evidences like paintings, municipal council minutes, project plans, construction permits, aerial views, photographs etc. Researchers and domain experts propose, discuss, debate different possible scenarios of urban evolution and propose various hypotheses. They use these evidences [Simon, 2012, Samuel et al., 2016] to study the existence [Hornsby and Egenhofer, 2000, Hallot and Billen, 2016] and non-existence (for e.g., destruction) of various city objects and to obtain a time duration for various features of city objects. These time-limited feature values of one or more city objects constitute a version[Van Ruymbeke et al., 2015, Chaturvedi et al., 2017, Van Ruymbeke et al., 2017]. However, there are also periods of time, where we do not have any evidence concerning the changes that may have occurred to these objects or the researchers may not be interested to know about these periods. Such gaps between two versions for the given objects is called a version transition [Chaturvedi et al., 2017]. A sequence [Van Ruymbeke et al., 2017, Van Ruymbeke et al., 2018] of versions and version transitions arranged in a chronological order of time may describe one of the possible ways a city could have evolved. However, multiple people working on the study either in isolation or in collaboration may propose different possible scenarios of changes based on the evidences available to them. Therefore, all these scenarios may be considered in parallel before a consensus is made on the best possible scenario. This final *consensus scenario* may be used as a reference by all the researchers until some other changes are decided upon. However, it is very important to save both the consensus scenario along with other proposed scenarios to understand other points of views on the urban changes and if necessary, to make more future modifications to the consensus scenario.

A scenario may also contain *imagined city versions* for representing and understanding other possible urban project plans proposed in the past that did not come to completion because of socio-economic or political pressures. Researchers want to understand how the city may have looked like if the proposed project were carried out. Documents like abandoned project plans, construction permits can give another possible view, detailing how the city would have looked like if the concerned plan had been carried out. New project plans often influence the surrounding areas especially leading to price fluctuations, construction of new habitat zones, shopping complexes and transport lines. In some cases, even abandoned projects are known [Samuel et al., 2016] to have left some traces on the cities, like in the case of Givors and Terrenoire, two old industrial towns of France [Périnaud et al., 2015]. Therefore, such an information of project influences on city infrastructure also forms a key constituent of urban evolution.

With the growing availability of different geographical information systems, the study of historical past has become a interdisciplinary field [Gregory and Cooper, 2013] and collaboration is important for discussing possible scenarios among stakeholders. In the past, researchers made use of physical models like plan-reliefs or virtual mockups for proposing their hypotheses. Physical

models are limited both in scale and temporal dimension and virtual mockups are created using the software technology and expertise available at the particular time of development making them unusable with the advancement of backward-incompatible technologies. Hence during the last few years, a data-driven approach is being considered, where standards are promoted by international communities in order to represent domain-specific knowledge. The primary goal of these standards is to promote data interoperability and easier sharing and exchange. Thus we require a solution that can support collaborative effort of the researchers using international standards able to represent 3D, temporal and domain-specific information concerning the cities.

In this article, we focus on representing possible scenarios of changes of a city or one or more city objects during a given period time. We believe that by adopting an abstract view for a generic approach, we free ourselves from specific technologies. Nevertheless, such an approach can be implemented using various existing software solutions. Our proposed approach called $UrbanCo^2Fab$ is a method that aims to provide the ability to represent several possible scenarios of urban evolution using interoperable data standards. It is a generic approach that can be targeted for several ongoing processes based on city models. Our proposition can be built over several existing software solutions. $UrbanCo^2Fab$ can also be extended to represent future imagined versions, but this requires further study and is out of scope for this paper. But this work can be seen as a base for studies related to scenarios of future urban evolution.

3 Related Works

We are interesting in reconstructing virtual cities of the past, both hypothetical and existing using a document corpus consisting of old postal cards, photographs, aerial views, newspaper archives, municipal meeting notes etc. Photographs of a building, for example at different periods of time are currently used by us to manually build virtual models of the building using standard city models at discrete time intervals. When extended to more buildings in a city sector or town, we get an overview of how a city evolved both horizontally and vertically. Evolution of cities [Renolen, 2000, Lefebvre et al., 2008, De Roo et al., 2013, Rollier-Hanselmann et al., 2014, Harbelot, 2015] has been studied in several different ways. Spatio-temporal object model (STOM) [Renolen, 2000] considers objects like roads, parcels along with their spatial, temporal and attribute changes and models both object-level and version-level transitions. These transitions are modeled using spatial, temporal, relationship and attribute descriptors. With the help of version-level transitions, it is possible to represent when an object ceased to exist, is tranformed from one form to another, is split into two different objects etc. Similarly, with the object-level transitions, it is possible to represent when an object is deduced or annexed from another object. OH_FET model [Lefebvre et al., 2008] also considers this object-oriented approach by considering the change in functional (usage), spatial and temporal information of the individual objects of interest. Continuum [Harbelot, 2015], a much recent work considers spatio-temporal changes as one single entity and tracks the structural, temporal and geometrical changes altogether. Though these studies are able to represent the version of one or more city objects at a given point of time, they focus on giving one possible hypothesis of the urban evolution. Additionally they do not consider imagined versions of urban evolution. Our proposed approach takes into consideration the above works and further explores the possibility of representing concurrent hypotheses: both factual and imagined. Finaly, custom-built versions or scenario of urban evolution with some proprietary technology or softwares is not our goal. In an interdisciplinary field, it has become very important to be able to exchange data in an interoperable manner. Most of the above works [Renolen, 2000, Lefebvre et al., 2008, Stefani et al., 2010, Simon, 2012, Harbelot, 2015] are not

based on any particular international standard. Nevertheless, these works form the basis for this proposed work.

An analogy of the proposed city object versions and scenarios can be made with the terminology used in management of software repositories. Software repositories (centralized and decentralized) [Otte, 2009] are commonly used in software development to promote parallel development by multiple programmers. Repositories and branches can be used by historians to suggest multiple possible historical successions of a city or city objects of interest. Yet current implementation of most of version control systems like GIT[1] is primarily targeted for text files and cannot be easily used to represent and understand the historical changes of city object features and to even propose any future imagined changes on them. We are interested in obtaining the value of features of such city objects at different temporal points or intervals. The focus on textual line changes and a missing object model make the existing implementations of software repositories unsuitable for studying historical evolution using standard city models, especially to study changes in city features or city object features. [Swierstra and Löh, 2014] recently formalized the key concepts of a version control systems, though once again focusing on line based version control systems. Nevertheless, by reutilizing the associated terminology in version control systems, the learning curve of the researchers can be reduced.

Finally in order to support interoperable solutions, we may base our approach on CityGML [Gröger et al., 2012] or city model [Stadler et al., 2009]. CityGML is an international standard proposed by OGC[2] for representing thematic, structural and semantic information of cities at different scales or levels of detail. It is also being currently used for the study of historical evolution [Pfeiffer et al., 2013, Billen et al., 2012, Morel and Gesquière, 2014, Samuel et al., 2016, Chaturvedi and Kolbe, 2019]. The standard [Tegtmeier et al., 2014, Chaturvedi and Kolbe, 2016] is currently being used by diverse communities for integrating domain specific information to the underlying city model[Gil et al., 2011]. It permits these extensions by Application Domain Extensions (ADE) that allows not only to easily share the extension but also the associated data in an interoperable manner. Take for example, a recent work studies various actions in understanding urban modeling [Sindram and Kolbe, 2014].

CityGML also considers the urban fabric from an object-oriented approach. In CityGML, for every city object, there is an associated information concerning its geometry and spatial position. CityGML 2.0 building model has two temporal information *yearOfConstruction* and *yearOfDemolition* for representing the year of construction and demolition respectively. However, this was very limited in scope since it could not be used to represent information on the real physical existence of other types of objects. With the latest proposed extension [Chaturvedi et al., 2017, Kutzner and Kolbe, 2018] to CityGML (see Figure 1), it is now possible to add a temporal dimension to every feature of the city object, i.e., a time validity for the existence of an object. This extension permits to represent lifetime of city objects right from its construction to its demolition and various changes in between, like change of roof structure of a building or its usage. However, this extension focused mainly on the physical existence of city objects. [Hallot and Billen, 2016] formalize the concept of existence of city objects and distinguish between existence and presence of objects. An object comes into existence as soon as a (semantic or spatial) relationship [Hallot and Billen, 2016] is established between this object and other objects under consideration. In our study, this notion of existence is important since we are interested in representing not only the physical presence of objects, but also their existence in the form of formal project plans or a thought process. We represent the changes to the objects during their existence time by giving a temporal validity to their known feature through $VersionableAbstractFeature$ and $Version$ (see Figure 1). The extension represents the features giving them a existence time

[1] https://git-scm.com/

[2] http://www.opengeospatial.org/

validity and groups these versionable features in user-defined versions. The users can then define the transitions between the versions detailing through the transactions (*Transaction*) of the various changes that occurred in between. One of their suggestions was to make use of the attribute tag in the *Version* (see Figure 1) to create agent-specific versions for various purposes like proposing historical evolution or even future scenarios. This suggestion enabled searching specific versions created by the agents with a given tag name to find a proposed historical evolution. But this proposition is very limited for practical uses involving a large number of researchers and it also missed a formal specification. Additionally their proposition also did not provide any constraint rules on the overall proposed model. Nevertheless, the above work has been extended in a much recent work [Samuel et al., 2016] to represent document and their references to different city objects. Taking into account these two works, we want to explore the study of urban evolution by making use of an approach commonly used in version control systems.

GeoGig[3], previously GeoGit [Clark et al., 2013] is one of the currently available software solutions that tackles the problem of tracking geospatial data changes made by users. However, it is not designed to handle situations where the agents wish to track changes of historical and future urban data. Besides it cannot currently handle CityGML files (or other city models). We have not found any solution that can meet all our above requirements of representing concurrent points of view of urban evolution, factual as well as planned and support for interoperable city data models. In particular, we want to not only obtain forward and backward traversal of the urban changes but also be able to see other possible propositions. In this article, we formalize our requirements and also present $UrbanCo^2Fab$ to fully describe our requirements. We may have several technical possibilities, but in this paper, we propose a formal approach that can be derived to a technical specification dedicated to a software or a format (e.g., GML, JSON etc.)

4 Workspaces for Concurrent City Projects

In this section, we introduce the concept of $Workspace$ and its formalization. A workspace manages various scenarios of urban evolution proposed by agents (e.g., researchers, users, domain experts, enterprises etc.). Like [Hallot and Billen, 2016], we consider a workspace consisting of one or more objects of interest and their associated period of existence. However, we wish to distinguish between the scenario that received a majority consensus and the scenarios that were proposed, debated or even abandoned. These scenarios, consisting of city object versions (both factual and imagined) enables representation of concurrent points of view for the study of historical urban evolution. We consider that every city object consisting of one or more geospatial features, each of which can be identified by an identifier [Hornsby and Egenhofer, 2000]. There could be multiple workspaces managed by independent groups working together in parallel. But in this article, we focus on formalizing a single workspace. An example workspace is given in Figure 2.

A workspace consists of two spaces: *Consensus Space* and *Proposition Space*. A space in a workspace lets a user create different scenarios of urban evolution. A *Consensus Space* is a mutually agreed upon space that can have only one scenario whereas the *Propositions Space* can have more than one scenarios. A scenario consists of one or more versions, where a version of urban evolution corresponds to the state of the city at a given instant of time (or time period). A scenario in our context is closely related to *interpretative sequence* proposed by [Van Ruymbeke et al., 2017, Van Ruymbeke et al., 2018], where the authors consider an *interpretative sequence* having multiple episodes (or versions). A version can be considered *Existing* if there is enough evidence for its materialized or physical existence [Hallot and Billen, 2016] in the real world.

[3] http://geogig.org

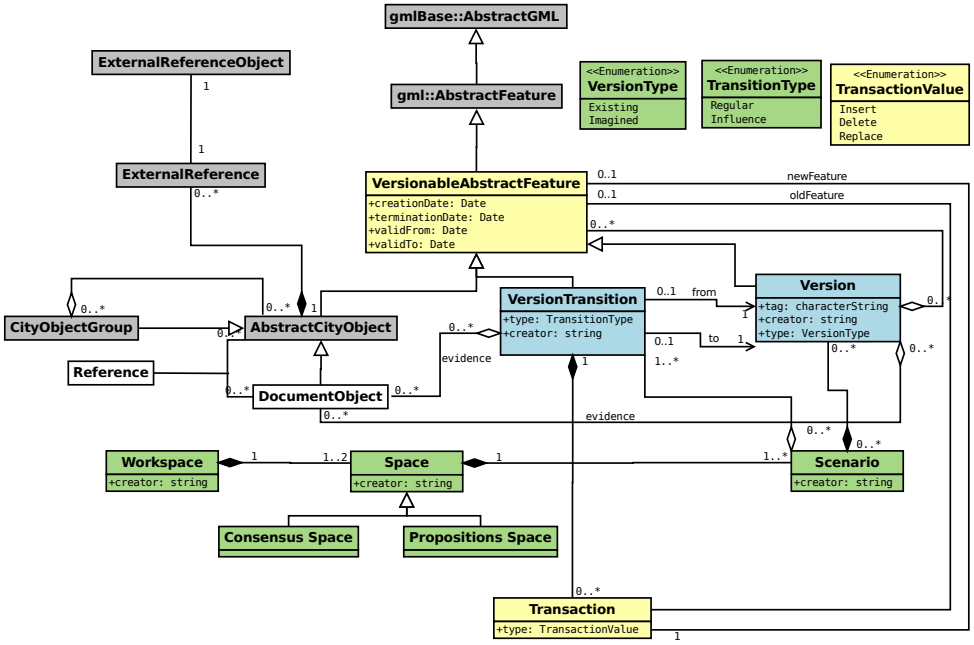

Fig. 1: UML model for representing concurrent points of view in CityGML. Grey boxes are existing classes of the CityGML core. Classes *VersionableAbstractFeature, TransactionValue, Transaction* represented by yellow-colored boxes are classes proposed to represent versioning of city objects [Chaturvedi et al., 2017]. *DocumentObject, Reference* represented by white-colored boxes are extensions proposed by [Samuel et al., 2016] to represent document references to objects. Classes *Version, VersionTransition* represented by blue-colored boxes are our proposed changes to the proposition by [Chaturvedi et al., 2017]. Classes *Workspace, Space, ConsensusSpace, PropositionsSpace, Scenario, VersionType, TransitionType* represented by green-colored boxes are newly proposed to support concurrent points of view in CityGML.

Imagined version corresponds to an imagined city (like in a urban plan) or a possible state (as proposed by a historian). Take for example, a photograph may be considered as sufficient evidence by a research group to call a version *Existing* and unimplemented project plans or mockups may be enough to call a version *Imagined*. Hence definition for *Existing* or *Imagined* versions is decided beforehand by a group based on the available categories of evidences. '*Transitions*' between versions are used to represent the change from one state to the other. A transition [Hornsby and Egenhofer, 2000] may signify a certain time period or point in time for which enough evidence on the city object(s) is not available. We also assume the ability to identify city

objects [Hornsby and Egenhofer, 2000] across different periods of time, i.e., we consider that it is possible to identify the objects that undergo changes. Take for example, we have two photographs (or videos) concerning the construction of a city building and its finished version, but we may not have any evidences on the stages of its development. This uncertain period is represented by a transition. As more evidence on intermediate states are obtained, it is possible to add new versions and transitions between them.

'*Influence*' is a special type of link between two versions in order to show that an imagined version (of the past) had some influence on a later version (*Existing*). This is used to represent situations where some project plans (like project plan for construction of a new highway) impact the city (like construction or destruction of houses).

A user would like to navigate in time to see the events at a given instant of time (both the past and the future with respect to that considered moment of time). As shown in Figure 2, the agent is at time T_{obs} and navigates the past and the future with respect to this point. We did not show the transactions in Figure 2. Transactions include addition, modification or removal of features of city object(s).

The key objects required to represent workspaces are further detailed in corresponding UML classes in the model of Figure 1. It is based on the extension of the CityGML core and slightly modifies the previous work [Chaturvedi et al., 2017] on the versioning of city objects. Yet the above model is not sufficient to specify the various constraint rules on each of the objects specified in the model. We now formalize workspaces using description logic, presenting both the key objects and the various constraint rules on them. It is important to note that on first look, it may seem that this work can be used immediately for imagining future scenarios starting from the current date. There are several questions that may require adjustments, especially with respect to transition types. Stating a statement on the current date like a future project influenced another future project is questionable. What evidences can be given to justify such a statement? These discussions are out of scope of this paper and will be presented in our future works.

4.1 Theoretical Preliminaries

We need to provide a formalized representation that can be mapped to several models that can manage city related information (e.g., CityGML). Additionally, we want to specify constraint rules on these models for representing a consistent city model. Current UML based model of CityGML (v2.0) doesn't specify the constraint rules. There are several ways of formalization like UML or description logic. We use description logic [Baader et al., 2003] to formalize the notions of concurrent points of view of urban evolution given its capability to define both the concepts and the associated constraints rules. This choice is also considering the fact that it can be easily mapped to semantic web languages and to ensure mapping to other possible data representations or city models. Let $\top$ be a top concept and $\bot$ be a bottom concept and A, B, $C, \ldots$ be atomic

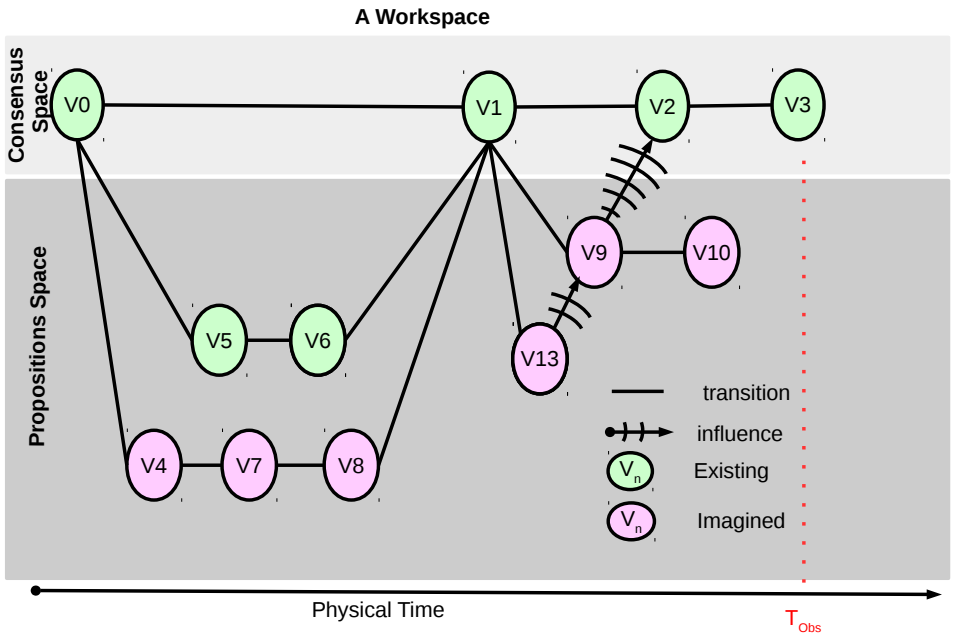

Fig. 2: Workspace for the study of historical evolution. There is one workspace with the consensus space and propositions space. A consensus space has only one scenario whereas the propositions space has multiple scenarios. Each scenario consists of many versions and transitions in between. There are two types of versions: Existing and Imagined. Two types of transitions are also shown: Regular transition (or simply Transition) and Influence. The agent is at time T_{Obs} and navigates through the time to study both the past and future with respect to T_{Obs}.

concepts. Let R be an atomic role. Following are also concepts:

$$\begin{aligned} \neg A &\text{ (negative concept)} \\ B \sqcap C &\text{ (intersection)} \\ B \sqcup C &\text{ (union of concepts)} \\ \forall R.C &\text{ (value restriction)} \\ \exists R.\top &\text{ (limited existential quantification)} \\ \exists R.C &\text{ (full existential quantification)} \\ \geq nR &\text{ (at-least number restriction)} \\ \leq nR &\text{ (at-most number restriction)} \\ = nR &\text{ (equal number restriction)} \end{aligned} \tag{1}$$

The interpretation of the above syntax is given by $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is called the domain of the interpretation and $.^{\mathcal{I}}$ is an assignment of every atomic concept A to a subset of $\Delta^{\mathcal{I}}$ and every atomic role R to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. ϕ corresponds to an empty set. $Delta^{\mathcal{I}} \backslash A^{\mathcal{I}}$ below corresponds to all elements present in $Delta^{\mathcal{I}}$ excluding those in Delta $A^{\mathcal{I}}$.

$$
\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \phi \\
(\neg A) &= \Delta^{\mathcal{I}} \backslash A^{\mathcal{I}} \\
(B \sqcap C)^{\mathcal{I}} &= B^{\mathcal{I}} \cap C^{\mathcal{I}} \\
(B \sqcup C)^{\mathcal{I}} &= B^{\mathcal{I}} \cup C^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \forall b.(a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\} \\
(\exists R.\top)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}}\} \\
(\exists R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \\
(\geq nR)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid |\{b|(a,b) \in R^{\mathcal{I}}\}| \geq n\} \\
(\leq nR)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid |\{b|(a,b) \in R^{\mathcal{I}}\}| \leq n\} \\
(= nR)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid |\{b|(a,b) \in R^{\mathcal{I}}\}| = n\}
\end{aligned} \tag{2}
$$

With a given interpretation $\mathcal{I}$, *inclusion* is defined in the following manner

$$B \sqsubseteq C, \text{ if and only if } B^{\mathcal{I}} \subseteq C^{\mathcal{I}} \tag{3}$$

Finally, *equivalence* is defined in the following manner

$$B \equiv C, \text{ if and only if } B^{\mathcal{I}} = C^{\mathcal{I}} \tag{4}$$

An equivalence is called a definition when the left side of the equation is an atomic concept (e.g., see definitions of $TemporalInterval$, $TemporalPoint$ etc.)

4.2 Time

In order to consider precise instants or time intervals, we introduce temporal point $TemporalPoint$ (a precise instant of the chosen timeline) and time interval $TimeInterval$ (period between two instants of the chosen timeline)[Harbelot, 2015].

$$TemporalPoint \sqsubseteq \top \tag{5}$$

A time interval defined by $[t_s, t_e]$, where $t_s, t_e \in TemporalPoint^{\mathcal{I}}$ correspond to start and end temporal points following strict ordering.

$$
\begin{aligned}
TimeInterval &\sqsubseteq \top \\
TimeInterval &\equiv (= 1 hasStartingPoint.TemporalPoint) \sqcap \\
&\quad (= 1 hasEndingPoint.TemporalPoint)
\end{aligned} \tag{6}
$$

We require τ to refer to temporal points or a time interval (e.g. documents like photographs are momentary whereas videos, project plans etc. have a time duration).

$$\tau \equiv TemporalPoint \sqcup TimeInterval \tag{7}$$

Temporal points and time intervals are disjoint.

$$TemporalPoint \sqcap TimeInterval \sqsubseteq \bot \tag{8}$$

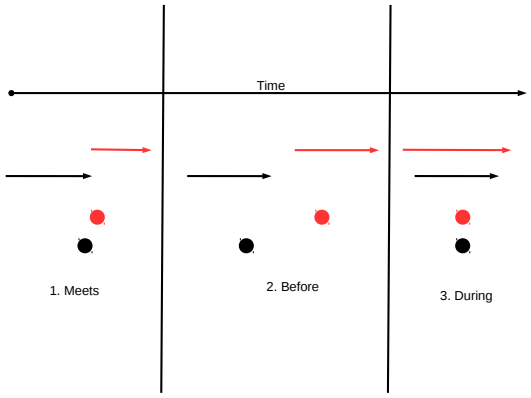

Fig. 3: Different time constraints: Dots represent temporal points and line represents time intervals. Three terms are introduced: *Meets, Before, During*. *Meets* shows that black colored temporal events are immediately followed by red-colored temporal events. *Before* corresponds to represent events when black colored temporal events are followed by red-colored temporal events but not immediately, i.e., there is a temporal gap of more than one instance of time between them. *During* shows that black colored temporal events occur at the same time as those of red-colored temporal events.

In order to specify various conditions like an instant (or a period) is immediately followed by another instant (or a period), occurred before another instant (period) or happened at the same instant (or during a given period), we introduce *meets* similarly like [Harbelot, 2015], $before$ and *during* respectively, inspired from [Allen, 1983] (See Figure 3).

$$meets^{\mathcal{I}} = \{< \mathcal{T}_i, \mathcal{T}_{i+1} > | i \in \mathbb{N}\} and \begin{cases} if\ \mathcal{T}_i, \mathcal{T}_{i+1} \in TemporalPoint \\ \forall \mathcal{T}_i, \mathcal{T}_{i+1} \rightarrow succ(\mathcal{T}_i) = \mathcal{T}_{i+1} \\ \\ if\ \forall a, b \in TemporalPoint^{\mathcal{I}} \\ if\ \forall I, I' \in TimeInterval^{\mathcal{I}} \\ if\ \forall (I, a) \in hasEndingPoint \\ if\ \forall (I', b) \in hasStartingPoint \\ \rightarrow a = b \end{cases} \tag{9}$$

$$before^{\mathcal{I}} = \{< \mathcal{T}_i, \mathcal{T}_{i+1} > | i \in \mathbb{N}\} and \begin{cases} if\ \mathcal{T}_i, \mathcal{T}_{i+1} \in TemporalPoint \\ \forall \mathcal{T}_i, \mathcal{T}_{i+1} \rightarrow \mathcal{T}_i < \mathcal{T}_{i+1} \\ \\ if\ \forall a, b \in TemporalPoint^{\mathcal{I}} \\ if\ \forall I, I' \in TimeInterval^{\mathcal{I}} \\ if\ \forall (I, a) \in hasEndingPoint \\ if\ \forall (I', b) \in hasStartingPoint \\ \rightarrow a < b \end{cases} \tag{10}$$

$$during^{\mathcal{I}} = \{< \mathcal{T}_i, \mathcal{T}_{i+1} > | i \in \mathbb{N}\} and \begin{cases} if\ \mathcal{T}_i, \mathcal{T}_{i+1} \in TemporalPoint \\ \forall \mathcal{T}_i, \mathcal{T}_{i+1} \rightarrow \mathcal{T}_i = \mathcal{T}_{i+1} \\ \\ if\ \forall a, b, c, d \in TemporalPoint^{\mathcal{I}} \\ if\ \forall I, I' \in TimeInterval^{\mathcal{I}} \\ if\ \forall (I, a) \in hasStartingPoint \\ if\ \forall (I, b) \in hasEndingPoint \\ if\ \forall (I', c) \in hasStartingPoint \\ if\ \forall (I', d) \in hasEndingPoint \\ \rightarrow a \geq c \wedge b \leq d \end{cases} \tag{11}$$

4.3 Versionable Features

In this section, we make use of the UML model given in [Chaturvedi et al., 2017], which in turn is based on the city model [Gröger et al., 2012] to define versionable abstract features.

Every feature is versioned by the duration of time interval. *VersionableAbstractFeature* is a time-stamped feature giving the agents the ability to specify the time duration (or validity) of the concerned feature. The relationship between versions and versionable features is shown with a graphical view in Figure 4. We consider two objects $O1$ and $O2$ composed of features like *name*, $function$, *geometry* and *storeysaboveground* and their changes in a particular period of time. As can be seen, whenever there is a change in some features like values, a version transition is created. Note the change in values for *storeysaboveground* and the resulting creation of versions and version transitions.

We introduce concepts $Name$ to define names and *Agent* to define agents responsible for creating or manipulation of instances.

In addition to an identifier [Hornsby and Egenhofer, 2000], every feature also has a name. This identification of feature is important to track every change that occured to any given feature.

$$\begin{aligned} AbstractFeature \equiv (= 1 hasName.Name) \sqcap \\ (= 1 hasIdentifier.Identifier) \end{aligned} \tag{12}$$

We now introduce $VersionableAbstractFeature$, that is an abstract feature with additional attributes to specify the existence of the feature.

$$VersionableAbstractFeature \sqsubseteq AbstractFeature \tag{13}$$

It has two properties $hasExistenceTime$ and $hasTransactionTime$ for representing the time period or interval of the existence of an object and the transaction interval concerning the addition of a object in a data store (e.g., database transaction interval).

$$\begin{aligned} VersionableAbstractFeature \equiv (= 1 hasExistenceTime.\tau) \sqcap \\ (= 1 hasTransactionTime.TimeInterval) \end{aligned} \tag{14}$$

An abstract city object [Chaturvedi et al., 2017] is a $VersionableAbstractFeature$.

$$AbstractCityObject \sqsubseteq VersionableAbstractFeature \tag{15}$$

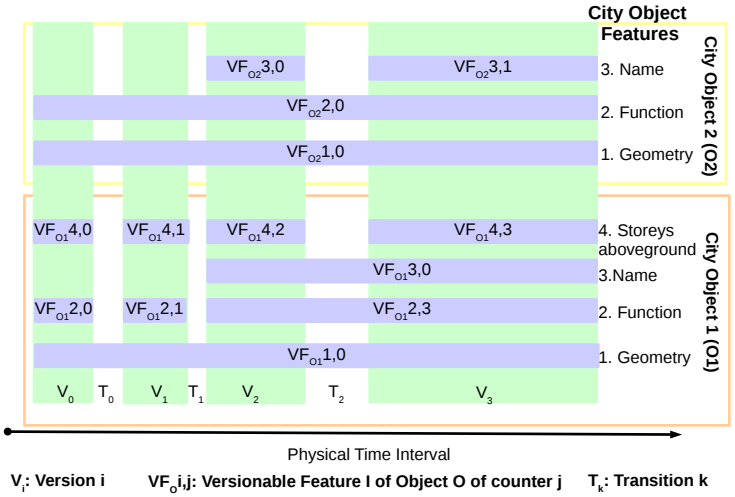

Fig. 4: Versions and versionable features. Box with yellow and orange colored strokes correspond to the features of object O_2 and O_1. A version can be defined as a set of stable version features of objects during the defined time interval of the version. The space between two consective versions corresponds to a version transition.

It is also possible to create groups of city objects. This *CityObjectGroup* inherits all the properties of AbstractCityObject. It also permits aggregation of city objects.

$$\begin{aligned} CityObjectGroup &\sqsubseteq AbstractCityObject \\ CityObjectGroup &\equiv (\geq 0 hasAbstractCityObject.AbstractCityObject) \sqcap \\ &\quad (= 1 hasParent.AbstractCityObject) \end{aligned} \tag{16}$$

It is possible to create city object groups of city object groups, the hierarchy of which is tracked by *hasParent*. *hasParent* is a transitive property. In order to prevent cycling of groups, the following condition is introduced (see also [Gröger et al., 2012]):

$$\forall o_c \forall o_p \forall o_a (hasParent(o_c, o_p) \wedge hasParent(o_d, o_a)) \rightarrow o_a \neq o_c \tag{17}$$

We introduce *DocumentObject* which is also considered as an abstract city object by [Samuel et al., 2016] and therefore inherits all its properties. A document object references other city objects (or another document, or collection of documents).

$$\begin{aligned} DocumentObject &\sqsubseteq AbstractCityObject \\ DocumentObject &\equiv (\geq 0 hasReference.AbstractCityObject) \end{aligned} \tag{18}$$

A document may concern another city object for a particular time instant (e.g., photos, newspapers) or an interval (e.g., videos), which is defined by the following definition of $hasReference$

$$hasReference \equiv (= 1 hasReferenceTime.\tau) \tag{19}$$

Changes occur during the course of time and these changes are captured by transactions. There are three ways a new transaction may occur: new features are added, values of features

may be changed or an old feature may be removed. Hence $TransactionValue$ is used to represent these three types of transactions:

$$\begin{aligned} TransactionValue \equiv Insert \sqcup \\ Replace \sqcup Delete \end{aligned} \tag{20}$$

A transaction [Chaturvedi et al., 2017] is defined to capture these changes between two temporal geospatial features.

$$\begin{aligned} Transaction \equiv (= 1 hasTransactionValue.TransactionValue) \sqcap \\ (\leq 1 hasOldVersionableAbstractFeature.VersionableAbstractFeature) \sqcap \\ (\leq 1 hasNewVersionableAbstractFeature.VersionableAbstractFeature) \end{aligned} \tag{21}$$

Insert: When a previously unknown value for feature is now known, Insert is used.

$$\begin{aligned} \forall t(hasTransactionValue(t, Insert) \\ \wedge hasOldVersionableAbstractFeature(t, NULL) \\ \wedge hasNewVersionableAbstractFeature(t, vf) \\ \wedge vf \neq NULL \end{aligned} \tag{22}$$

Note that $NULL$ can be used to signify unknown value or no value.

Delete: When a previously known value for a feature is no longer valid and a new value is not available anymore, Delete is used.

$$\begin{aligned} \forall t(hasTransactionValue(t, Delete) \\ \wedge hasOldVersionableAbstractFeature(t, vf) \\ \wedge hasNewVersionableAbstractFeature(t, NULL) \\ \wedge vf \neq NULL \end{aligned} \tag{23}$$

Replace: When a previously known value for a feature is no longer valid and a new value is available, Replace is used.

$$\begin{aligned} \forall t(hasTransactionValue(t, Replace) \\ \wedge hasOldVersionableAbstractFeature(t, vf_1) \\ \wedge hasNewVersionableAbstractFeature(t, vf_2) \\ \wedge hasExistenceTime(vf_1, tm_1) \wedge hasExistenceTime(vf_2, tm_2) \\ \wedge before(tm_1, tm_2) \wedge vf_1 \neq vf_2) \end{aligned} \tag{24}$$

4.4 Version

A version is created by a user and is a collection of temporal geospatial features [Chaturvedi et al., 2017], geospatial features with a timespan. There are two types of versions: *Existing* or *Imagined*. For every version, there may exist some supporting evidence.

$$\textbf{VersionType} \equiv Existing \sqcup Imagined \tag{25}$$

A version can be considered Existing based on the materialized evidence available to the user to prove the real physical existence. It is entirely left to the researchers in how they use the

document evidences to decide whether a version is imagined or existing. A version may have user-defined tags.

$$\begin{aligned} \textbf{Version} &\sqsubseteq VersionableAbstractFeature \\ \textbf{Version} &\equiv (=1 hasVersionType.VersionType) \sqcap \\ &(\geq 0 hasEvidence.DocumentObject) \sqcap \\ &(\geq 0 hasTag.Tag) \sqcap \\ &(\geq 1 hasVersionableAbstractFeature.VersionableAbstractFeature) \sqcap \\ &(=1 hasCreator.Agent) \end{aligned} \tag{26}$$

The existence time of a version must be contained within the existence time of every versionable abstract feature contained in it.

$$\begin{aligned} \forall v \forall vf (&hasVersionableAbstractFeature(v, vf) \wedge hasExistenceTime(v, tm_v) \\ &hasExistenceTime(vf, tm_f) \\ &\rightarrow during(tm_v, tm_f)) \end{aligned} \tag{27}$$

4.5 Version Transitions

A version transition [Chaturvedi et al., 2017] is a collection of transactions between two versions, i.e., it a collection of changes between two versions. There are two types of version transitions: *Regular* and Influence. Regular transitions are transitions used to represent the time period (or interval) between two consecutive versions.

$$\begin{aligned} \textbf{TransitionType} \equiv &Regular \sqcup \\ &Influence \end{aligned} \tag{28}$$

$$VersionTransition \sqsubseteq VersionableAbstractFeature \tag{29}$$

$$\begin{aligned} \textbf{VersionTransition} \equiv &(\geq 0 hasEvidence.DocumentObject) \sqcap \\ &(\geq 1 hasTransaction.Transaction) \sqcap \\ &(=1 hasPreviousVersion.Version) \sqcap \\ &(=1 hasNewVersion.Version) \sqcap \\ &(=1 hasTransitionType.TransitionType) \sqcap \\ &(=1 hasCreator.Agent) \end{aligned} \tag{30}$$

Following condition ensures that an Influence transition type is allowed between two versions (for example, in Figure 2, a transition between $V9$ to $V2$), where the previous version is imagined and the new version is existing and also the imagined version must exist before the existing version.

$$\begin{aligned} \forall vt (&hasTransitionType(vt, Influence) \wedge \\ &hasPreviousVersion(vt, v_p) \wedge hasNewVersion(vt, v_n) \wedge \\ &hasVersionType(v_p, Imagined) \wedge hasVersionType(v_n, Existing) \wedge \\ &hasExistenceTime(v_p, tm_1) \wedge hasExistenceTime(v_n, tm_2)) \rightarrow before(tm_1, tm_2) \end{aligned} \tag{31}$$

Following conditions ensure that a Influence transition type is allowed between two imagined versions having different existence times (one before the other) as in Figure 2, an example transition between $V13$ to $V9$.

$$\begin{aligned} \forall vt(hasTransitionType(vt, Influence) \wedge \\ hasPreviousVersion(vt, v_p) \wedge hasNewVersion(vt, v_n) \wedge \\ hasExistenceTime(v_p, tm_1) \wedge hasExistenceTime(v_n, tm_2)) \rightarrow before(tm_1, tm_2) \end{aligned} \tag{32}$$

The time limits of the version transition must be greater than or equal to the end time of old version and the start time of the new version

$$\begin{aligned} \forall vt(hasTransitionType(vt, Influence) \wedge \\ hasPreviousVersion(vt, v_p) \wedge hasNewVersion(vt, v_n) \wedge \\ hasVersionType(v_p, Imagined) \wedge hasVersionType(v_n, Imagined) \wedge \\ hasExistenceTime(v_p, tm_1) \wedge hasExistenceTime(v_n, tm_2) \wedge \\ hasExistenceTime(v_p, tm)) \rightarrow meets(tm_1, tm) \wedge meets(tm, tm_2) \end{aligned} \tag{33}$$

4.6 Scenario

A scenario [Chaturvedi et al., 2017] or an interpretative sequence [Van Ruymbeke et al., 2017, Van Ruymbeke et al., 2018] consists of a number of versions.

$$\begin{aligned} \textbf{Scenario} \equiv (\geq 1 hasVersion.Version) \sqcap \\ (\geq 0 hasVersionTransition.VersionTransition) \sqcap \\ (= 1 hasTransactionTime.TimeInterval) \sqcap \\ (= 1 hasCreator.Agent) \end{aligned} \tag{34}$$

All the version transitions present in a scenario also must have their one previous and next version existing in the scenario.

$$\begin{aligned} \forall vt(hasVersionTransition(b, vt) \wedge \\ hasPreviousVersion(vt, vt_p) \wedge hasNewVersion(vt, vt_n)) \\ \rightarrow hasVersion(b, vt_n) \wedge hasVersion(b, vt_p) \end{aligned} \tag{35}$$

For any two versions present in the scenario, there must exist a version transition.

$$\begin{aligned} \forall v_1 \forall v_2 \exists vt(hasVersionTransition(b, vt) \wedge \\ hasVersion(b, v_1) \wedge hasVersion(b, v_2) \\ \wedge hasPreviousVersion(vt, v_2) \wedge hasNewVersion(vt, v_1)) \end{aligned} \tag{36}$$

There must not be any existence time overlap between any two versions or any two version transitions.

$$\begin{aligned} \forall v_1 \forall v_2 (hasVersion(b, v_1) \wedge hasVersion(b, v_2) \\ hasExistenceTime(v_1, tm_1) \wedge hasVersion(v_2, tm_2)) \\ \wedge tm_1 = tm_2 \rightarrow v_1 = v_2 \end{aligned} \tag{37}$$

No two version transitions in a branch can have the same existence time.

$$\begin{aligned}\forall vt_1 \forall vt_2 (hasVersionTransition(b, vt_1) \wedge hasVersionTransition(b, vt_2) \\ hasExistenceTime(vt_1, tm_1) \wedge hasExistenceTime(vt_2, tm_2)) \\ \wedge tm_1 = tm_2 \rightarrow vt_1 = vt_2\end{aligned} \tag{38}$$

There exists only one version that is shared between the two transitions, i.e., the next version of a version transition is the same as previous version of another one.

$$\begin{aligned}\exists vt_1 \forall vt_2 (hasVersionTransition(b, vt_1) \wedge hasVersionTransition(b, vt_2) \\ hasNewVersion(vt_1, v_1) \wedge hasPreviousVersion(vt_2, v_1))\end{aligned} \tag{39}$$

These conditions are necessary to ensure a single sequence of changes proposed by a scenario.

4.7 Workspace for concurrent points of view

A workspace has two spaces: *Consensus Space* and *Propositions Space*. Both these spaces contain scenarios detailing the changes occuring in a given place or time through a sequence of versions. A *Consensus Space* is the reference space used by the researchers in a group and there is a consensus among them about its contents. A *Consensus Space* has only one scenario whereas the*Propositions Space* can have zero or more scenarios. Contrary to the *Consensus Space*, a *Propositions Space* is used by agents to propose alternate scenarios.

$$Space \sqsubseteq \top \tag{40}$$

$$\begin{aligned}\textbf{Space} \equiv (\geq 1 hasTransaction.Transaction) \sqcap \\ (= 1 hasCreator.Agent)\end{aligned} \tag{41}$$

Propositions Space has zero or more scenarios. A scenarios in*Propositions Space* may have different type of versions having version type Existing or Imagined.

$$\begin{aligned}\textbf{PropositionsSpace} \sqsubseteq Space \\ \textbf{PropositionsSpace} \equiv (\geq 0 hasScenario.Scenario)\end{aligned} \tag{42}$$

Consensus Space has a scenario with all the versions having version type Existing.

$$\begin{aligned}\textbf{ConsensusSpace} \sqsubseteq Space \\ \textbf{ConsensusSpace} \equiv (= 1 hasScenario.Scenario)\end{aligned} \tag{43}$$

All the version types in a *Consensus Space* must be Existing.

$$\begin{aligned}\forall v (ConsensusSpace(c) \wedge hasScenario(c, b) \wedge hasVersion(b, v) \\ \wedge hasVersionType(v, t_v)) \rightarrow t_v = Existing\end{aligned} \tag{44}$$

A workspace consists of one *Consensus Space* and one *Propositions Space*. It has an owner.

$$\begin{aligned}\textbf{Workspace} \equiv (= 1 hasConsensusSpace.ConsensusSpace) \sqcap \\ (= 1 hasPropositionSpace.PropositionsSpace) \sqcap \\ (= 1 hasTransactionTime.TimeInterval) \sqcap \\ (= 1 hasCreator.Agent)\end{aligned} \tag{45}$$

Consensus Space and*Propositions Space* are mutually disjoint.

$$ConsensusSpace \sqcap PropositionsSpace \sqsubseteq \bot \tag{46}$$

Consensus Space and*Propositions Space* must share at least one common version (i.e., the starting point).

$$\begin{aligned} \exists v_{b_c}, \exists v_{b_p} (ConsensusSpace(c) \wedge hasScenario(c, b_c) \wedge hasVersion(b_c, v_{b_c}) \\ PropositionsSpace(p) \wedge hasScenario(p, b_p) \wedge hasVersion(b_p, v_{b_p})) \\ \rightarrow v_{b_c} = v_{b_p} \end{aligned} \tag{47}$$

5 *UrbanCo²Fab*: Collaborative Comprehension of Urban Fabric

In section 4, we presented a new formalization to manage concurrent points of view of urban evolution. These rules pave way to manage all the possible use cases and are used to develop the $UrbanCo^2Fab$ command-line tool [Samuel et al., 2018]. It can represent concurrent points of view of the evolution of one or more city objects proposed by a group of users by ensuring the consistency of the underlying city data and its extensions by making use of rules. The command line application focuses on management of workspaces and is based on the above proposed rules and CityGML. CityGML uses identifiers to identify the city objects and uses XML data format. Every CityGML city object has a set of attributes and associated values. Thanks to the XML format and path query languages like XPath, it is possible to get the value of any attribute of a city object given its object identifier. Considering the close similarity between workspaces and current version control systems, we proposed the extensions [Samuel et al., 2018] required to implement $UrbanCo^2Fab$ using git version control system. To reduce the learning curve of $UrbanCo^2Fab$ command line tool, we reused the vocabulary of git like *init, commit, push, pull, add, rm* [Samuel et al., 2018].

As described above in section 3, current implementations of GIT mainly targets line based changes. In our case, we are interested to know the city objects that underwent a change between two versions along with the changes in their attribute values. In order to work with CityGML files (XML files), we needed to modify the way differences between two changes of a file. $UrbanCo2Fab$ command line tool, developed using python library Pygit2[4] uses git to store both the CityGML files and the key metadata of versions, version transitions, scenarios and workspace. A user writes to a CityGML file the values of different attributes of relevant city objects with their identifiers. Then the user commits a version (using the command *urbanco2fab commit* [Samuel et al., 2018]) assigning a physical existence time for a given version. The identifiers of a (git) commit are used to identify versions. User can continue creating new versions for specifying changes to city objects. To see the changes between any two versions, the classical *git diff* tool is not very useful since it will show the lines that changed. However, *urbanco2fab diff* makes use of the *git diff* to not only identify these lines but also the city objects that underwent changes. Thanks to this, the user can see the object (identifiers) as well as the attribute value changes (replace, insert etc.)

There are three major metadata files in $UrbanCo2Fab$: *workspace.json*, *scenarios.json* and *versions.json*. Recall that the identifier of a commit in GIT is a unique hash value generated taken into account the transaction time, the commit message, the author and the diff. A version transition is identified by two version identifiers, i.e., if id_1 and id_2 are two identifiers of a commit (or a version), a possible version transition can be identified by $id_1 - id_2$. The version metadata

[4] http://pygit2.org/

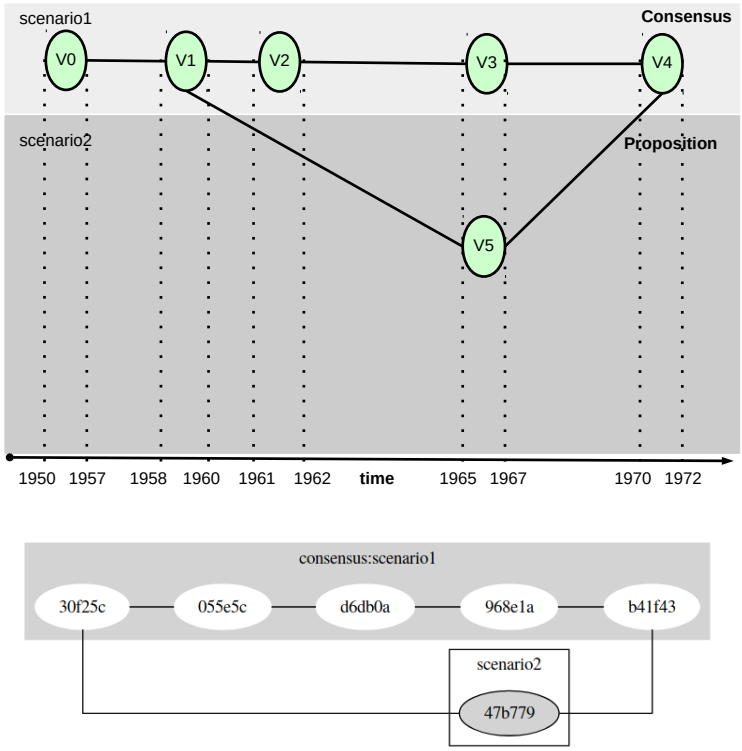

Fig. 5: Example of workspace and output of *urbanco2fab viz* showing one consensus scenario (scenario1) and one proposition scenario (scenario2). 6 first letters of hash created following a *urbanco2fab commit* are used to identify the different versions

file *versions.json* is used to track the changes to the city model files (CityGML). It contains important information related to every version: identifier, title, description, related documents, physical existence time of a given version, version type, transaction times etc. The scenario metadata files *scenarios.json* store the key information concerning the different scenarios created by users, the associated versions and version transitions, the transactions of version transitions, title, description, scenario type etc. The workspace metadata file *workspace.json* contains the identifiers of consensus and proposition scenarios.

Figure 5 shows the output of *urbanco2fab viz*. It shows two scenarios in workspace: one consensus and one proposition scenario. The 6-character identifier is the version identifier (6 first letters of hash created following a *urbanco2fab commit*). Internally, $UrbanCo2Fab$ uses pygit2 to generate version identifiers. The consensus scenario has five versions *30f25c, 055e5c, d6db0a, 968e1a, b41f43* and the proposition scenario *scenario2* has three versions *30f25c, 47b779, b41f43*. Thanks to the constraint rules presented in section 4, it is possible to verify validity of a version, a version transition and a scenario. These rules are used to reject cases like when a user tries to create scenarios with versions having the same physical existence times etc.

Fig. 6: UDV: Application to visualize and navigate 3D cities

6 Conclusion

Study of urban evolution of a city helps not only to understand its history but also to obtain valuable lessons from the past for its future plans. The work started by [Chaturvedi et al., 2017] to represent temporal evolution of city objects was enhanced in this article to represent concurrent possible hypotheses of urban evolution. Our new contributions on *Workspace, Scenario, Version, Version Transition* and *Versionable Features* enable researchers to represent, store and navigate multiple parallel hypotheses on the evolution of a city. Based on these enhancements and the initial proposal from [Chaturvedi et al., 2017], we defined a complete set of rules in order to describe the urban evolution. We developed a proof of concept $UrbanCo^2Fab$ by making use of the formalization rules linked with real city models. Analogous to existing control systems and their well-known options, we tend to reduce the learning curve of the end users of our proposal. Our next goal is to fully integrate our solution to the current 3D visual environment so that users can easily visualize the urban past. Figure 6 shows the current 3D visualization tool called UDV [Gaillard et al., 2015]. Additonal work is needed to link UDV with our current proposal on $UrbanCo^2Fab$ to allow complete navigation in 4D urban environment. This will be completed by providing a graph-based visual interactive tool that can guide the users in the concurrent views of city lifecycle. Another objective will be to manage additional rules for rights management on the data related to workspaces and versions.

Acknowledgements

This work was performed within the framework of the LABEX IMU (ANR-10-LABX-0088) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007) operated by the French National Research Agency (ANR). We would like to thank Clémentine Périnaud, Vincent Jaillot, Eric Boix, Frédéric Pedrinis and Jérémy Gaillard for their feedback. We would also like to thank anonymous reviewers for their valuable feedback, thanks to which we we were able to improve our work.

Bibliography

References

[Allen, 1983] Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843.

[Baader et al., 2003] Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

[Billen et al., 2012] Billen, R., Carré, C., Delfosse, V., Hervy, B., Laroche, F., Lefevre, D., Servieres, M., and Van Ruymbeke, M. (2012). 3d historical models: the case studies of liege and nantes. In *COST Action TU801 workshop on Semantic Enrichment of 3D city models for sustainable urban development*.

[Chaturvedi and Kolbe, 2016] Chaturvedi, K. and Kolbe, T. H. (2016). Integrating dynamic data and sensors with semantic 3d city models in the context of smart cities. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W1:31–38.

[Chaturvedi and Kolbe, 2019] Chaturvedi, K. and Kolbe, T. H. (2019). A requirement analysis on extending semantic 3d city models for supporting time-dependent properties. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-4/W9:19–26.

[Chaturvedi et al., 2017] Chaturvedi, K., Smyth, C. S., Gesquière, G., Kutzner, T., and Kolbe, T. H. (2017). Managing versions and history within semantically enriched 3d city models. *Advances in 3D Geoinformation, Lecture Notes in Cartography and Geoinformation, Springer*.

[Clark et al., 2013] Clark, S., Mesdaghi, S., Palumbo, D., and Holmes, C. (2013). Rapid open geospatial user-driven enterprise (rogue) joint capability technology demonstration white paper.

[De Roo et al., 2013] De Roo, B., Bourgeois, J., and Maeyer, P. D. (2013). On the way to a 4d archaeological gis: state of the art, future directions and need for standardization. *Proceedings of the 2013 Digital Heritage International Congress. Vol. 2.*, page .

[Fields, 2012] Fields, G. (2012). Urbanization and the Transition from Agrarian to Industrial Society. *Berkeley Planning Journal*, 13(1).

[Gaillard et al., 2015] Gaillard, J., Vienne, A., Baume, R., Pedrinis, F., Peytavie, A., and Gesquière, G. (2015). Urban data visualisation in a web browser. In Jia, J., Hamza-Lup, F. G., and Schreck, T., editors, *Proceedings of the 20th International Conference on 3D Web Technology, Web3D 2015, Heraklion, Greece, June 18-21, 2015*, pages 81–88. ACM.

[Gil et al., 2011] Gil, J., Almeida, J., and Duarte, J. P. (2011). The backbone of a city information model (cim): Implementing a spatial data model for urban design. In *29th eCAADe Conference, Ljubljana, Slovenia, 21-24 September 2011*.

[Gregory and Cooper, 2013] Gregory, I. and Cooper, D. (2013). Geographical Technologies and the Interdisciplinary Study of Peoples and Cultures of the Past. *Journal of Victorian Culture*, 18(2):265–272.

[Gröger et al., 2012] Gröger, G., Kolbe, T. H., C., N., and K. H., H. (2012). OGC city geography markup language (CityGML) encoding standard v2.0. *OGC Doc*, page .

[Hallot and Billen, 2016] Hallot, P. and Billen, R. (2016). Enhancing spatio-temporal identity: States of existence and presence. *ISPRS International Journal of Geo-Information*, 5(5):62.

[Harbelot, 2015] Harbelot, B. (2015). *Continuum : a spatio-temporal and semantic model for the discovery of dynamic phenomena within geospatial environments*. PhD thesis, University of Burgundy, Dijon, France.

[Harbelot et al., 2013] Harbelot, B., Arenas, H., and Cruz, C. (2013). Continuum: a spatiotemporal data model to represent and qualify filiation relationships. In Kashani, F. B., Basalamah, A., and Zhang, C., editors, *Proceedings of the 4th ACM SIGSPATIAL International Workshop on GeoStreaming, IWGS 2013, November 5, 2013, Orlando, FL, USA*, pages 76–85. ACM.

[Hornsby and Egenhofer, 2000] Hornsby, K. and Egenhofer, M. J. (2000). Identity-based change: a foundation for spatio-temporal knowledge representation. *International Journal of Geographical Information Science*, 14(3):207–224.

[Kutzner and Kolbe, 2018] Kutzner, T. and Kolbe, T. H. (2018). Citygml 3.0: Sneak preview. In Kersten, T. P., Glch, E., Schiewe, J., Kolbe, T. H., and Stilla, U., editors, *PFGK18 - Photogrammetrie - Fernerkundung - Geoinformatik - Kartographie, 37. Jahrestagung in Mnchen 2018*, volume 27 of *Publikationen der Deutschen Gesellschaft fr Photogrammetrie, Fernerkundung und Geoinformation (DGPF) e.V.*, pages 835–839, Mnchen. Runder Tisch GIS e.V.; Deutsche Gesellschaft fr Photogrammetrie, Fernerkundung und Geoinformation (DGPF) e.V.; Deutsche Gesellschaft fr Kartographie (DGfK) e.V., Deutsche Gesellschaft fr Photogrammetrie, Fernerkundung und Geoinformation e.V.

[Lefebvre et al., 2008] Lefebvre, B., Rodier, X., and Saligny, L. (2008). Understanding urban fabric with the oh_fet model based on social use, space and time. *Archeologia e calcolatori 19*, pages 195–214.

[Morel and Gesquière, 2014] Morel, M. and Gesquière, G. (2014). Managing temporal change of cities with citygml. *Eurographics Workshop on Urban Data Modelling and Visualisation*, pages 37–42.

[Otte, 2009] Otte, S. (2009). Version control systems. *Computer Systems and Telematics*, page .

[Périnaud et al., 2015] Périnaud, C., Gay, G., and Gesquière, G. (2015). Exploration of the changing structure of cities: Challenges for temporal city models. In *2015 Digital Heritage*, volume 2, pages 73–76.

[Peuquet and Wentz, 1994] Peuquet, D. and Wentz, E. (1994). *An approach for time-based analysis of spatiotemporal data*, pages 489–504. Taylor & Francis, London.

[Pfeiffer et al., 2013] Pfeiffer, M., Carré, C., Delfosse, V., Hallot, P., and Billen, R. (2013). Virtual leodium: from an historical 3d city scale model to an archeological information system. *ISRP Annals of Photogrammetry, 2-5/W1*.

[Renolen, 2000] Renolen, A. (2000). Modelling the real world: Conceptual modelling in spatiotemporal information system design. *Trans. GIS*, 4(1):23–42.

[Rollier-Hanselmann et al., 2014] Rollier-Hanselmann, J., Petty, Z., Mazuir, A., Faucher, S., and Coulais, J.-F. (2014). Développement d'un sig 4d pour la ville médiévale de cluny. *Archeologia e calcolatori*, pages 164–179.

[Samuel et al., 2016] Samuel, J., Prinaud, C., Servigne, S., Gay, G., and Gesquire, G. (2016). Representation and Visualization of Urban Fabric through Historical Documents. In Catalano, C. E. and Luca, L. D., editors, *Eurographics Workshop on Graphics and Cultural Heritage*. The Eurographics Association.

[Samuel et al., 2018] Samuel, J., Servigne, S., and Gesquière, G. (2018). Urbanco2fab: Comprehension of concurrent viewpoints of urban fabric based on git. volume IV-4/W6, pages 65–72.

[Simon, 2012] Simon, G. (2012). Modélisations multi-scalaires des dynamiques urbaines dans la longue durée: l'exemple du quartier abbatial de vendôme (41). *Cybergeo: European Journal of Geography*.

[Sindram and Kolbe, 2014] Sindram, M. and Kolbe, T. H. (2014). Modeling of urban planning actions by complex transactions on semantic 3d city models. In *Proceedings of the 7th International Congress on Environmental Modelling and Software*.

[Stadler et al., 2009] Stadler, A., Nagel, C., König, G., and Kolbe, T. H. (2009). Making interoperability persistent: A 3d geo database based on citygml. In *3D Geo-Information Sciences*, pages 175–192. Springer.

[Stefani et al., 2010] Stefani, C., De Luca, L., Véron, P., and Florenzano, M. (2010). Time indeterminacy and spatio-temporal building transformations: an approach for architectural heritage understanding. *International Journal on Interactive Design and Manufacturing (IJIDeM)*, 4(1):61–74.

[Swierstra and Löh, 2014] Swierstra, W. and Löh, A. (2014). The semantics of version control. In Black, A. P., Krishnamurthi, S., Bruegge, B., and Ruskiewicz, J. N., editors, *Onward! 2014, Proceedings of the 2014 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software, part of SPLASH '14, Portland, OR, USA, October 20-24, 2014*, pages 43–54. ACM.

[Tegtmeier et al., 2014] Tegtmeier, W., Zlatanova, S., van Oosterom, P. J. M., and Hack, H. R. G. K. (2014). 3d-gem: Geo-technical extension towards an integrated 3d information model for infrastructural development. *Computers & Geosciences*, 64:126–135.

[Van Ruymbeke et al., 2015] Van Ruymbeke, M., Carré, C., Delfosse, V., Pfeiffer, M., and Billen, R. (2015). Towards an archaeological information system: Improving the core data model. In *CAA 2014 21st century Archaeology: Concepts methods and tools: Proceedings of the 42nd Annual Conference on Computer Applications and Quantitative Methods in Archaeology*, pages 245–253. Archaeopress.

[Van Ruymbeke et al., 2017] Van Ruymbeke, M., Hallot, P., and Billen, R. (2017). Enhancing cidoc-crm and compatible models with the concept of multiple interpretation. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W2:287–294.

[Van Ruymbeke et al., 2018] Van Ruymbeke, M., Hallot, P., Nys, G.-A., and Billen, R. (2018). Implementation of multiple interpretation data model concepts in CIDOC CRM and compatible models. *Virtual Archaeology Review*, 9(19):50.