

A new nD Temporal Geodata Management Approach using GML

Martin Hoppen, Michael Schluse, Juergen Rossmann

Christoph Averdung

Institute for Man-Machine Interaction
RWTH Aachen University
Aachen, Germany

CPA Geo-Information
Siegburg, Germany
Email: averdung@supportgis.de

Email: {hoppen, schluse, rossmann}@mmi.rwth-aachen.de

Abstract—Geodata represents the state of a real world phenomenon at some point in time. Multiple, differently interpreted reference times can be associated to the same datum yielding an n -time-dimensional (nD) problem. This reference time, however, is often ignored, reduced or managed "manually" on application level. We propose a new approach for nD temporal geodata management encapsulating and hiding this complexity. The Geography Markup Language (GML) and a transactional Web Feature Service (WFS-T) are used for standard compliant data exchange. Its feasibility is proven by a reference implementation and applications. Thus, the main achievement of this paper is the presentation of a new nD temporal data management approach using time point representation and its application to GML-based forestry applications.

Keywords—Geodata Management; Temporal Data Management; Temporal Databases; GML; WFS-T.

I. INTRODUCTION

In geodata, the concept of time is pervasive. Geodata represents the state of a real world phenomenon at one or more points in time. Orthogonally, the data itself may be captured, manually revised, or updated at different times. There may also be some point in time where the geodata becomes effective, i.e., valid in a formal sense. Finally, when incorporating geodata into simulation models, simulation time can be considered, as well. These different interpretations of time values are called *time dimensions* [1].

Thus, when dealing with geodata, its time references can only be omitted in the simplest cases. An appropriate temporal geodata management is therefore advisable. However, managing temporal references may be a complex, tedious and error-prone task – in particular when performed "manually" on application level. Instead, one should prefer a transparent data management layer, encapsulating and hiding this complexity. Another advantage of such an application-independent approach is that the same management layer can be reused for different applications. Regarding the aforementioned diversity of time dimensions, the underlying temporal modeling concept should furthermore be sufficiently universal.

In previous publications, we presented the fundamentals for using temporal data management in 3D simulation applications [2]. Subsequently, we mentioned the idea of an n -time-dimensional versioning concept for geodata [3]. In this paper, we introduce a detailed mathematical specification of this concept and demonstrate its feasibility by means of a reference implementation on geodatabase and on client application level. Furthermore, we show how this higher-order

temporal metadata is marshaled into a GML representation for data exchange – in particular based on WFS-T. Finally, the approach is used as a basis for distributed parallel data management.

The rest of the paper is structured as follows. In Section II, we give an overview of the state of the art. In the following section (Section III), the new concept for nD -versioning is introduced. Its realization in a geodatabase system and integration into a 3D simulation system is presented in Sections IV–V. The approach is applied to a distributed parallel data management scenario in Section VI presenting a corresponding application in Section VII. We conclude the paper in Section VIII.

II. STATE OF THE ART

A. Temporal Databases

As described in [4][5], in general, a temporal database is any database containing some temporal information. This can be explicitly modeled within the scope of the application schema, e.g., a date field "start of employment" for an employee's record. Here, application programs have to manage temporal data on their own, which can be a complex task for sophisticated scenarios. To cope with this complexity, different concepts have been developed generalizing the problem.

Furthermore, a time value must be interpreted. Such interpretations are called time dimensions [1]. The time associated with an event can represent its occurrence in the real world. This time dimension is called *valid time*, the corresponding database is called a *valid time database*. When time values refer to the point in time when the values were stored in the database, it is called *transaction time*, the corresponding database *transaction time database*. A database combining both time dimensions is a bitemporal database. Besides these two most common time dimensions there can be more interpretations.

When using valid time, for each operation (read or write) the user has to supplement the point in real time corresponding to the event. This also allows to specify future or past events leading to so-called *proactive* or *retroactive updates*. When using transaction time, however, timestamps for writing operations are automatically derived from the database's system clock; proactive or retroactive updates are impossible. For reading operations omitting to specify a transaction time, the current (i.e., last written) value is returned. This can also be specified as a transaction time of *cur* (current). For other points in time, the corresponding historic value is returned.

When updating a data item in a temporal database, its previous state (i.e., version) is not lost. Rather, a new version with the new value is inserted and the previous version is closed becoming a so-called history version. Thus, every data item is represented by a set of $n > 0$ versions. The state of a temporal database at a certain point in time (or a tuple thereof) is called a snapshot. Likewise, non-temporal databases are called snapshot databases.

A usual approach is to store start as well as end timestamps for each version and time dimension. While every version must have start timestamps, end times are only set to close a version. Deleting a data item is only performed logically by closing its current version. Inserting a new data item creates an initial version. Following this standard approach, in bitemporal databases, an update even creates two new versions. By updating a data item, its current version is closed concerning valid time by setting an end timestamp. The timestamp's update itself however leads to a new version regarding transaction time but still holding the previous value of the data item. The second version is created with the actual new value and current timestamps.

In relational databases, this concept can be implemented by adding start and end timestamp attributes for each time dimension to every relation. This has the drawback of redundancy as changing a single attribute value of a tuple creates a new version duplicating all other attribute values. This can, for example, be reduced by dividing the relation into subrelations. In object-oriented or object-relational databases, a common approach is attribute-based versioning. Here, usually, an attribute version is stored as a tuple of start and end time for each time dimension and the corresponding value. Thus, for each of an object's attributes, a list of such tuples is stored. In addition, the object itself maintains so-called *lifespan* temporal attributes for each time dimension. They represent its overall lifetime from creation to deletion.

Following [6], the aforementioned approach of storing start as well as end times for each version is called an *interval representation*. In contrast, storing only start times is called a *time point representation*. Here, subsequent versions' start times close preceding versions. Consequently, each version has a certain *scope* comprising those points in time the version's value is "visible."

An extensive overview of existing spatio-temporal database models is given in [7]. While about 2/3 of the systems are bitemporal, in contrast to the proposed approach, none provides three (or more) time dimensions. Furthermore, only about 1/3 of the proposed models have been implemented and about 1/3 lack a formal representation. Other, more recent approaches like [8] do integrate more dimensions (scale) but reduce time to one dimension. Finally, a very recent publication [9] presents a substantial online bibliography on various aspects of temporal GIS, which has yet to be examined.

B. Temporal Data Management Standards

The International Organization for Standardization (ISO) norm 19100 [10] series of geographic information standards in combination with the specifications of the Open Geospatial Consortium (OGC) [11] provide the basis for designing the structure of time-related geospatial data. For a standard-based

design of geospatial data structures with a focus on database-driven data management, the following ISO standards are important:

- ISO 19119: Geographic information - Services
- ISO 19109: ... - Rules for application schema
- ISO 19107: ... - Spatial schema
- ISO 19111: ... - Spatial referencing by coordinates
- ISO 19125: ... - Simple feature access
- ISO 19136: ... - Geography Markup Language (GML)

ISO 19108 adds specifications for the description of time-related issues. This is the basis for extending the usually 3-dimensional structure of geospatial reference systems by temporal properties. For that purpose, ISO 19108 provides rules for the schema design of time as an absolute (point of time, duration) or a chronological (before, after, in-between) specification of time.

The geospatial data's reference time can be modeled in different ways, depending on the technical requirements of the given task. It can either be part of the data's schema itself (*explicit modeling*) or it can be a built-in property of the applications system architecture (*implicit modeling*). In both cases, ideally, the specifications of ISO 19108 are taken into account yielding an implementation that considers international standards.

III. nD-VERSIONING

As motivated in Section I, besides valid and transaction time (e.g., combined in a bitemporal database), even more time dimensions can be thought of. One of these is the effectivity of a change, i.e., the time it becomes effective or valid in a formal sense. For example, in data acquisition for a forestry application (compare Section VII), tree heights may be updated over a longer period of time. Each height value is associated with the appropriate valid timestamp (actual time of the measurement) and transaction timestamp (time of the database entry). The third timestamp represents the time a change becomes effective, e.g., the beginning of the next inventory period at the end of all measurements. This allows to decouple the description of real world phenomena with valid time from the effectivity of these values while modeling the more or less technical information about data storage with transaction time. While this three-time-dimensional scenario shall be used to motivate and explain the following specification, our concept can be generalized to arbitrary kinds and numbers of time dimensions.

For this higher-order temporal database, a new concept for evaluating time point representations was developed. As mentioned above, in this case, historic versions are only implicitly closed by other versions' timestamps. We show how this representation can be applied to all integrated time dimensions at the same time.

A. One and Two Time Dimensions

To begin with, this approach can also be applied for one or two time dimensions. In the former case, every update is associated with a single (scalar) timestamp t_i with associated version v_i . The interval $[t_i, t_j)$, where $t_j > t_i$ is the next timestamp, defines v_i 's scope (with $t_j = \infty$ if t_i is the last timestamp). A retroactive update (e.g., in case of a valid time database) with timestamp t_k , $t_i < t_k < t_j$ splits the existing interval into $[t_i, t_k)$ and $[t_k, t_j)$.

In the bitemporal case (i.e., with the two time dimensions transaction and valid time), an update is associated with a timestamp tuple (t_i^T, t_i^V) for transaction time T and valid time V . Here, the two-time-dimensional scope of the associated version v_i initially (without any other existing versions) is a quarter-plane restricted by the two rays emerging from (t_i^T, t_i^V) parallel to and in positive direction of the respective axis (dotted area in Figure 1 left).

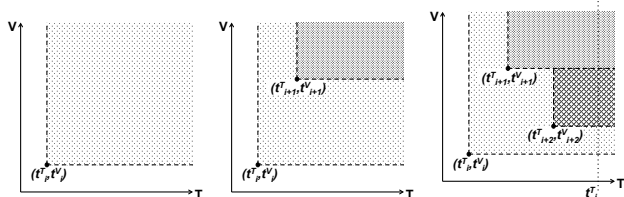


Figure 1. Bitemporal example of the evaluation schema for the higher-order temporal database.

Inserting a new version v_{i+1} with a timestamp tuple (t_{i+1}^T, t_{i+1}^V) with $t_i^T < t_{i+1}^T, t_i^V < t_{i+1}^V$ (Figure 1 center) reduces the scope of v_i to two infinite "tubes" while the new scope of v_{i+1} is again defined by a quarter-plane rooted at (t_{i+1}^T, t_{i+1}^V) . Thus, value version v_{i+1} is only visible when considering a reference time greater or equal in both timestamp components. As transaction time does not allow retroactive updates, $t_i^T < t_{i+1}^T$ must always be valid. However, valid time allows for $t_i^V > t_{i+1}^V$. An example is shown in Figure 1 (right). A new version v_{i+2} with timestamp tuple (t_{i+2}^T, t_{i+2}^V) is inserted where $t_{i+1}^T < t_{i+2}^T, t_{i+1}^V > t_{i+2}^V$. While v_i 's scope is similarly restricted as in the previous example, v_{i+2} 's scope stops at v_{i+1} 's. We call this behavior the "Golden Rule": The expansion of a limiting ray stops at preexisting other scopes. In the exemplary scenario, the expansion of the ray in V -direction (V -ray) shall stop. This allows retroactive updates with regard to valid time to "slide in between" existing values. That is, for any $t^T \geq t_{i+2}^T$, the scope of v_{i+2} is between v_i 's and v_{i+1} 's scopes. As a corollary, T -rays do *not* stop at existing V -rays. In particular, this makes the scope of new versions independent of their insertion's order regarding valid time. In Figure 1 (right), the order of the valid timestamp components is irrelevant to the resulting partitioning along the V axis. Figure 2 gives an example for an alternate insertion order. The partitioning regarding valid time along any $t_j^T > t_{i+2}^T$ is identical to the prior case.

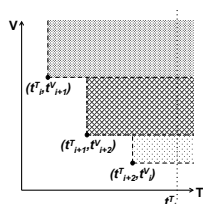


Figure 2. Inserting versions in an alternate order (i.e., different transaction times) yields the same partitioning along a $t_j^T > t_{i+2}^T$.

B. Three Time Dimensions

This evaluation schema can now be extended to the three time dimensions as mentioned above: transaction time T ,

valid time V , and effectivity time E . This leads to three-dimensional timestamps (t_i^T, t_i^V, t_i^E) for every version v_i . The resulting scope (to begin with, without interference of other timestamps) is a "cube" with infinite extent in direction of each axis. The simplest case can be compared to the 2D case in Figure 1 (center) and is shown in Figure 3: The scope for timestamp (t_i^T, t_i^V, t_i^E) is reduced to (up to) three "walls" by a new version with timestamp tuple $(t_{i+1}^T, t_{i+1}^V, t_{i+1}^E)$ if $t_{i+1}^T \geq t_i^T, t_{i+1}^V \geq t_i^V, t_{i+1}^E \geq t_i^E$. Note that – as for the 2D case – the actual scopes infinitely expand in (positive) axis directions.

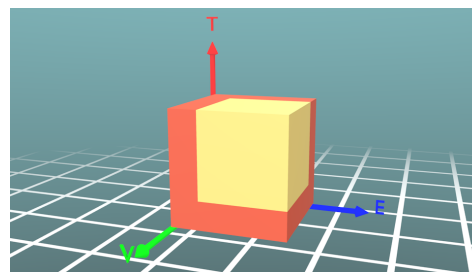


Figure 3. Intersection of i^{th} (orange) and $(i + 1)^{th}$ (yellow) 3D time scope for $t_{i+1}^T \geq t_i^T, t_{i+1}^V \geq t_i^V, t_{i+1}^E \geq t_i^E$.

For other constellations, again, the "Golden Rule" resolves ambiguities. In the 3D case, the rays emerging from a timestamp tuple in each axis direction have to be tested against the orthogonal quarter-planes defined by the other two axes of other timestamps. An example is given in Figure 4. The V -ray of one timestamp (orange) is tested against the quarter-plane defined by the T - and E -rays (T - E -quarter-plane) of a second timestamp (yellow). To allow for retroactive updates relative to the V -axis to "slide in between" existing values as in the two-time-dimensional case, the V -ray has to stop at the T - E -quarter-plane.

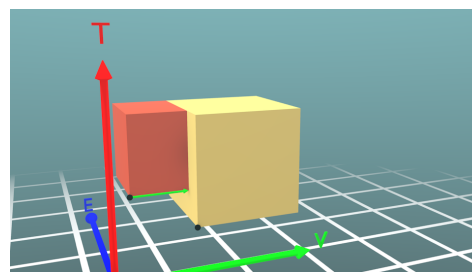


Figure 4. Expansion in V -direction of one timestamp's scope (orange) stops at T - E -quarter-plane of other timestamp's scope (yellow).

In general, the rule has to state whether a T -, V -, or E -ray has to stop at the respective orthogonal V - E -, T - E -, or T - V -quarter-planes. Like in the 2D case, a desired property is that the partitioning shall be independent of the versions' insertion order, i.e., of their timestamps' T component. Corresponding to the partitioning of the vertical line at t_j^T in Figure 1 (right) and Figure 2, this is the partitioning of a "high enough" intersecting V - E -plane, i.e., for a t_j^T greater than any occurring t_i^T . An example is given in Figure 5. Compared to Figure 4, the T component t_i^T of the i^{th} timestamp (orange) is less than that of t_{i+1}^T (yellow). In each case, the respective V and E components are identical. In both scenarios, however, the

final (i.e., with regard to T) partitioning along an intersecting V - E -plane (depicted in purple in Figure 5) is equal. In the first scenario ($t_i^T > t_{i+1}^T$, Figure 4), a V -ray *must* stop at a T - E -quarter-plane. In the second ($t_i^T < t_{i+1}^T$), an E -ray *must not* stop at a T - V -quarter-plane. Otherwise, the partitioning would not be consistent.

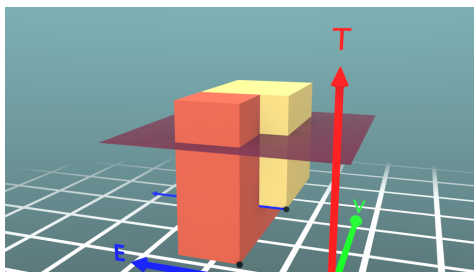


Figure 5. Expansion of E -ray must not stop at T - V -quarter-plane to yield the same partitioning on a V - E -plane (purple) as in Figure 4.

Figures 4 and 5 depict the case where $t_i^V < t_{i+1}^V$ and $t_i^E > t_{i+1}^E$. For reasons of symmetry, the same arguments apply to the case where $t_i^V > t_{i+1}^V$ and $t_i^E < t_{i+1}^E$. In fact, only the indexes (i and $i + 1$) and thus the colors of the two scopes (orange and yellow) need to be swapped.

The other distinct case is $t_i^V < t_{i+1}^V$ and $t_i^E < t_{i+1}^E$. For $t_i^T < t_{i+1}^T$ this is already depicted in Figure 3. Here, all coordinates of timestamp i (orange) are less than those of timestamp $i + 1$ (yellow). To achieve the same partitioning on a V - E -plane for $t_i^T > t_{i+1}^T$, T -rays *must not* stop at V - E -quarter-planes. Figure 6 shows an example. The T -ray emitted from the $(i + 1)^{th}$ timestamp (yellow) does *not* stop at the V - E -quarter-plane defined by the i^{th} timestamp (orange). The dashed ovals mark the yellow scopes expansion reducing the orange scope. Again, the same argument applies to $t_i^V > t_{i+1}^V$ and $t_i^E > t_{i+1}^E$ and can be depicted by switching colors (orange and yellow) in Figures 5–6.

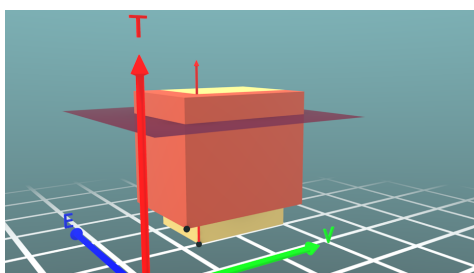


Figure 6. Expansion of a T -ray must not stop at V - E -quarter-planes to yield the same partitioning on a V - E -plane (purple) as in Figure 3.

Altogether, the Golden Rule for the three-time-dimensional case states that

- T -rays *must not* stop at V - E -quarter-planes,
- V -rays *must* stop at T - E -quarter-planes, and
- E -rays *must not* stop at T - V -quarter-planes.

The constraints for the choice of these rules are

- partitioning on a "high enough" V - E -plane shall be independent from insertion order, i.e., transaction time T and

- retroactive updates relative to the V -axis shall "slide in between" existing values.

IV. IMPLEMENTATION IN GEODATABASE

Our goal was to implement the time-related behavior of a geospatial reference system independently of any manufacturer. This approach ensures the portability between different database systems like Oracle or PostgreSQL, especially when managing time-related geospatial data. For that purpose, during the development of the presented n D temporal data management system, a transformational layer was introduced as an agent between application development and physical data storage of geospatial data (Figure 7). This layer provides a transformation between the temporal properties of the data schema (independent of explicit or implicit modeling of time) and its implementation, e.g., in terms of tables in an object-relational database.

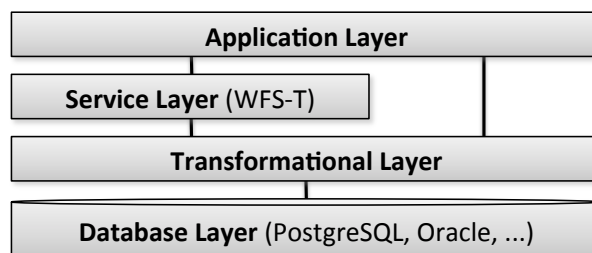


Figure 7. Layered architecture of the proposed approach.

On the application layer, GML is used to describe and request geospatial time-related data within the proposed data management system. A GML-encoded dataset combined with a request (select, insert, update, delete) is transformed into Structured Query Language (SQL) statements by the transformational layer.

The transformational layer is configured using an application schema provided as an Extensible Markup Language (XML) schema file (XML Schema Definition (XSD)) using the specifications of ISO 19109. This schema defines the rules for the automatic generation of SQL statements for the n D data storage.

For OGC-compliant data retrieval and manipulation, a Web Feature Service (WFS 2.0) was implemented as an interface to the transformational layer. It accepts geospatial data (in terms of GML 3.2.1) with multiple timestamps that is passed to the n D data management system where a consistent handling of the space-time-reference is ensured.

V. INTEGRATION INTO SIMULATION SYSTEM

As presented in [2], we supply our simulation system clients with a shared simulation model. While the model is managed by a central (geo)database, each client uses an in-memory runtime simulation database to locally cache the model for real-time access (Figure 8). This combination of databases can be seen as a distributed database system that uses change notifications for replication and synchronization [12].

The simulation database itself is a snapshot database (compare Subsection II-B). Thus, replication from the central geodatabase must be performed with reference to a certain

timestamp tuple provided by the user or an application component. To make the snapshot consistent, the entirety of replicate copies must always represent the same timestamp tuple. Thus, on changing the reference time, replication has to be updated. A naive approach is to unload all objects and then reload them with the new reference date. A more efficient approach is to only reload data that changed in between the previous and the new reference time. For this purpose, however, the central geodatabase must be able to (quickly) provide a list of all changes between two reference time tuples, e.g., by maintaining a queryable global change log.

Figure 8 shows an example scenario for the bitemporal case. The central geodatabase contains a simple forest model comprising a *Tree* class with a *height* attribute. A snapshot of the object *a : Tree* is replicated to the client using reference time ($t^T=cur, t^V=2012$). By using transaction time $t^T=cur$ (the default access mode for most users), the last written value for valid time $t^V=2012$ is replicated. Thus, the height of *a : Tree*'s snapshot within the simulation database is 16.0m.

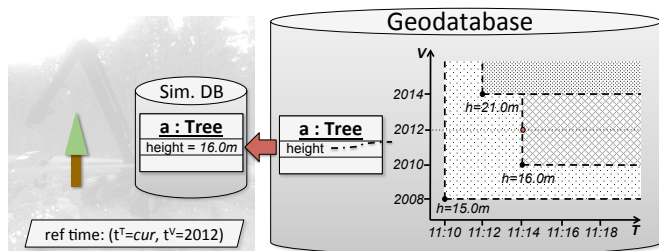


Figure 8. Bitemporal example: Replicating a snapshot of a tree object from a central geodatabase with reference time ($t^T=cur, t^V=2012$).

Besides replicating data from the central geodatabase, changes within a simulation client can also be synchronized back. For that purpose, first, the local snapshot must be refreshed to represent the desired reference time for the change. Subsequently, the change is applied locally and an update transaction with the exact same time tuple is issued to the central geodatabase. For the same reason, the simulation database must be considered read-only when the snapshot represents historic versions regarding transaction time (i.e., $t^T \neq cur$). Here, changes cannot not be resynchronized as transaction time does not support retroactive updates.

An example for updating the previously replicated *a : Tree* object is given in Figure 9.

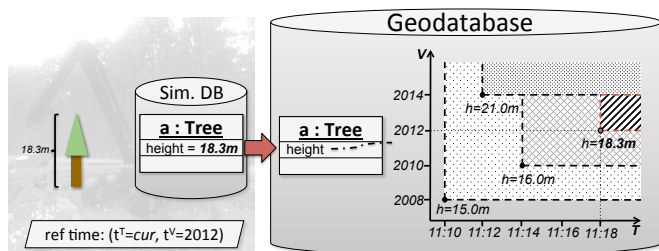


Figure 9. Bitemporal example: Updating a tree object's height within the central geodatabase for reference time ($t^T=cur, t^V=2012$) (transaction executed at 11:18h).

The tree's height actually measured 18.3m in 2012 so

it must be updated. While keeping the same reference time ($t^T=cur, t^V=2012$), the new value is resynchronized to the geodatabase. It is inserted with the chosen valid time $t^V=2012$ and the system's current time for transaction time – in this case $t^T=11:18$.

VI. DISTRIBUTED PARALLEL DATA MANAGEMENT

Based on the presented *nD*-versioning approach, a concept for distributed parallel data management was developed. In this scenario, the master copy of the considered geodata is managed in a central temporal database while replicate copies are distributed to several temporal working databases, on-demand. The latter are distributed over remote sites, e.g., at different contractors (or departments) of the main organization, with limited or no direct connectivity to the central site. A WFS-T (Figure 7) at the master site is used for data exchange. Note, however, that these working databases are *not* the simulation databases from Section V but rather additional temporal geodatabases.

For data acquisition (check-out), depending on connectivity, a contractor either directly queries the WFS-T or delegates his request to the organization. Given authorization, the data is marshaled to a GML representation comprising all its versions. In the GML representation, objects and properties are associated with timestamps in terms of XML attributes. To allow for XML validation, an extended schema specifying these attributes is issued by the WFS-T. On remote site, the GML data is unmarshaled (i.e., imported) into the working database. Altogether, this allows remote sites to access the full history of the acquired data – not only a snapshot. The contractor can perform his tasks on the data within his working database. To transmit his results back to the central site (check-in), changes are tracked within the working database. They are marshaled to a similar GML representation, which, however, only contains the changed objects, as well as the associated timestamps and their full history. This GML dataset is sent to the WFS-T – as before, either directly or indirectly – for check-in into the master database. Furthermore, the concept can also be applied in a cascade. Here, the contractor itself distributes the data once again to 2^{nd} tier working databases among his employees using a local WFS-T.

To allow different contractors to work in parallel on the same data, it can be replicated multiple times. For that purpose, it is marked with so-called *process control objects* within the central database. In contrast to locking, this allows for longterm processes to be performed in parallel without mutual exclusive access. Process control objects also define the reason for a check-out. Based on existing markers, further check-out requests can be granted for non-exclusive processes. This process-based approach allows for a flexible concurrency control within the distributed data management scenario. Due to this optimistic approach, however, change conflicts may arise when the same data item is changed by different contractors. Using the temporal information, this can be detected on check-in time by analyzing changes that occurred since the contractor's check-out by comparing the available full history of the changed object. Additionally, besides temporal data, metadata describing the job and the contractor is managed allowing to determine responsibilities for changes. For that purpose, before the actual check-in, a simulated check-in is performed: The corresponding check-out is repeated to a local

helper database where the contractor’s changes are applied. Here, conflicts can be detected and either be directly resolved or referred to the contractor for resolution. When no conflicts occur, the actual check-in is performed and the corresponding process control marker is removed.

VII. APPLICATIONS

The research project Virtual Forest [13][14] is one of the primary applications for the presented approach. One of the core ideas of this project is a consistent, shared data model and data management in the Virtual Forest database. Provided to all stakeholders in this field, it facilitates the exploitation of know-how and synergies. Furthermore, it supports the transfer of industrial automation techniques to the forest industry. Note, however, that the presented techniques are not restricted to this context. Figure 10 shows one of the realized applications using the presented approach. The user interface to set the reference times (“Referenzdatum”) is shown in detail. It allows to set transaction (“Systemzeit”), valid (“Stichtag”) and effectivity time (“Gültig ab”). Transaction time can be set to *current* (“jetzt”).

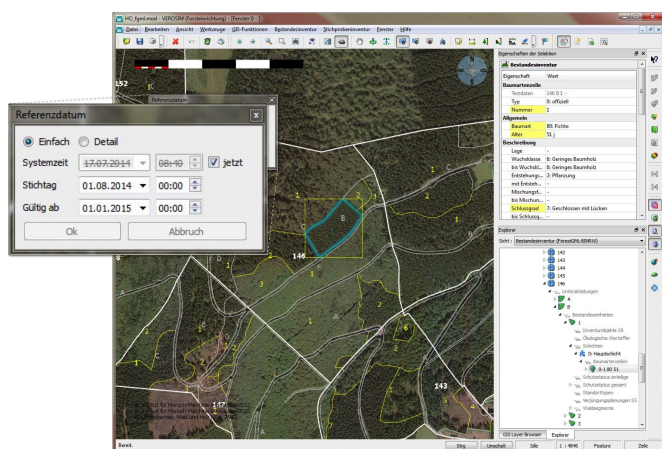


Figure 10. A forest inventory tool based on the presented approach - the reference date for the local snapshot can be specified by the user.

For the Virtual Forest database, currently, the three aforementioned time dimensions are utilized. Transaction time is used to model the technical change history. It allows to determine when forestal data was changed, thus, documenting work processes. The second time dimension is valid time. In particular for individual tree objects, it is used to directly model the actual age. Id est, a tree’s “birth” is modeled as a new tree object inserted with the corresponding valid time. Any change to its properties is equally updated, e.g., a height measured at a certain point in time (see example in Section V). In stand inventory, forests are not modeled as individual trees but in terms of so-called *stand units*. A stand unit object combines a surface geometry with aggregated properties like 75% spruce with average age of 80y and average height of 30m. All associated updates to this stand inventory data are typically bound to a certain valuation date within the considered year, e.g., October 1st. Thus, all updates use this date as valid time, although the actual values are typically recorded over a longer period of time around this date. Effectivity time is used to model the point in time when new values shall become the official representation of the forest. Updating huge forest

datasets using fieldwork or automated processes (as developed in the Virtual Forest project) can take a longer period around the valuation date. By using an effectivity time well after this period (e.g., next New Year’s Day), no intermediate results are visible to other users evaluating the currently effective data.

In the context of the Virtual Forest project, a GML application schema called ForestGML was developed to consistently model forest-related data. To add the necessary multiple timestamps on a per-attribute basis, standard GML or WFS concepts like `<wfs:FeatureCollection timeStamp="...">` do not suffice. Thus, to embed *n*D temporal reference information and corresponding metadata into this (or any other) application schema, it is extended by XML attributes. Figure 11 gives a simplified example of a stand unit object including all its historical attribute versions for a check-out process. The object itself is extended by the creation timestamps ($t^T=2010-06-01$, $t^V=2010-10-01$ and $t^E=2011-01-01$). The object’s creation is described in the metadata object referenced by `meta_start_id`. The tree species code and area percentage attribute both were set during object creation yielding the same reference time and metadata object. However, the percentage value was changed at a later point in time ($t^T=2014-06-01$, $t^V=2014-10-01$ and $t^E=2015-01-01$) in the context of another job described by `metadata2`.

```
<wfs:member>
  <fgml:Stand gml:id="object1"
    meta_start_id="metadata1"
    t_T="2010-06-01"
    t_V="2010-10-01"
    t_E="2011-01-01">
    <fgml:treeSpecies t_T="2010-06-01"
      t_V="2010-10-01"
      t_E="2011-01-01"
      meta_id="metadata1">80</fgml:treeSpecies>
    <fgml:percentage t_T="2010-06-01"
      t_V="2010-10-01"
      t_E="2011-01-01"
      meta_id="metadata1">80.0</fgml:percentage>
    <fgml:percentage t_T="2014-05-01"
      t_V="2014-10-01"
      t_E="2015-01-01"
      meta_id="metadata2">90.0</fgml:percentage>
    ...
  </fgml:Stand>
</wfs:member>
```

Figure 11. Simplified example using the ForestGML application schema extended by *n*D temporal information and metadata.

Finally, using the presented approach for distributed parallel data management, inventory jobs can be distributed among several contractors (or employees of one contractor). Processes modeled using process control objects comprise inventory jobs, geometry revisions, or annual automatic growth extrapolations. Occurring change conflicts, e.g., due to inaccurately performed jobs, can be resolved as described in Section VI.


VIII. CONCLUSION AND FUTURE WORK

All in all, we presented a novel approach to manage temporal geodata with multiple time dimensions. In particular, it exceeds the bitemporal scenario adding further flexibility in temporal data modeling. Furthermore, we developed a new approach to use point representations for several time dimensions by defining a “Golden Rule” to resolve ambiguities. Based on

this nD -versioning, a flexible approach for distributed parallel data management is presented. In the end, prototypical implementations and the practical usage for forestry applications already prove the feasibility of the approach.

As presented in [12], our approach for distributed database synchronization with multiple clients relies on change notifications. This concept has to be extended for time dimensions allowing retroactive or proactive updates. On the part of the central (geo)database, notifications would need to comprise the timestamp tuple corresponding to the notified change. The component for distributed synchronization, in turn, would need to consider these timestamps. The particular notification handling strategy requires further research. Likewise, for the multiple time dimension case, the efficient updating of local replicate copies within a simulation database when changing the considered reference time has to be analyzed.

ACKNOWLEDGMENT

Virtual Forest:  This project is co-financed by the European Union and the federal state of North Rhine-Westphalia, European Regional Development Fund (ERDF). Europe - Investing in our future.

REFERENCES

- [1] "The Consensus Glossary of Temporal Database Concepts," URL: <http://people.cs.aau.dk/~csj/Glossary/index.html> [accessed: 2015-01-03].
- [2] M. Hoppen, M. Schluse, J. Rossmann, and B. Weitzig, "Database-Driven Distributed 3D Simulation," in Proceedings of the 2012 Winter Simulation Conference, 2012, pp. 1–12.
- [3] J. Rossmann, A. Bücken, and M. Hoppen, "Semantic World Modelling and Data Management in a 4D Forest Simulation and Information System," ISPRS 8th 3DGeoInfo Conference & WG II/2 Workshop, International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. XL-2/W2, 2013, pp. 65–72.
- [4] R. Elmasri and S. B. Navathe, Database Systems: Models, Languages, Design, And Application Programming, 6th ed. Prentice Hall International, 2010.
- [5] T. Myrach, Temporale Datenbanken in betrieblichen Informationssystemen, 1st ed. Wiesbaden: Teubner Verlag, 2005.
- [6] J. Clifford and A. U. Tansel, "On an algebra for historical relational databases: two views," ACM SIGMOD Record, vol. 14, no. 4, May 1985, pp. 247–265.
- [7] N. Pelekis, B. Theodoulidis, I. Kopanakis, and Y. Theodoridis, "Literature Review of Spatio-temporal Database Models," The Knowledge Engineering Review, vol. 19, no. 3, Sep. 2004, pp. 235–274.
- [8] P. van Oosterom and J. Stoter, "5D data modelling: full integration of 2D/3D space, time and scale dimensions," in Proceedings of the 6th international conference on Geographic information science. Zurich, Switzerland: Springer-Verlag, Sep. 2010, pp. 310–324.
- [9] W. Siabato, C. Claramunt, M. A. Manso-Callejo, and M. A. Bernabé-Poveda, "Time Bliography: A Dynamic and Online Bibliography on Temporal GIS," Transactions in GIS, vol. 18, no. 6, Dec. 2014, pp. 799–816.
- [10] ISO/TC 211 Geographic information/Geomatics, "ISO 19100," URL: <http://www.isotc211.org> [accessed: 2015-01-03].
- [11] "Open Geospatial Consortium (OGC)," URL: <http://www.opengeospatial.org> [accessed: 2015-01-03].
- [12] M. Hoppen and J. Rossmann, "A Database Synchronization Approach for 3D Simulation Systems," in DBKDA 2014, The 6th International Conference on Advances in Databases, Knowledge, and Data Applications, A. Schmidt, K. Nitta, and J. S. Iztok Savnik, Eds., Chamonix, France, 2014, pp. 84–91.
- [13] J. Rossmann, M. Schluse, and A. Bücken, "The virtual forest - Space and Robotics technology for the efficient and environmentally compatible growth-planing and mobilization of wood resources," FORMEC 08 - 41. Internationales Symposium, 2008, pp. 3 – 12.
- [14] J. Rossmann, M. Schluse, R. Waspe, and R. Moshhammer, "Simulation in the Woods: From Remote Sensing based Data Acquisition and Processing to Various Simulation Applications," in Proceedings of the 2011 Winter Simulation Conference, S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, Eds., 2011, pp. 984 – 996.