Definition and Implementation of Temporal Operators for a 4D Query Language

S. Daum and A. Borrmann

Chair of Computational Modeling and Simulation, Technische Universität München, Arcisstraße 21, 80333 München; PH +49 89 289-23047; FAX +49 89 289 25051; email: {daum,borrmann}@bv.tum.de

ABSTRACT

Currently, there is no technology available to validate 4D Building Information Models using formal methods of temporal and spatial analysis. We propose to fill this technological gap by providing a query language which provides dedicated spatial and temporal operators. In the presented approach, a building is digitally represented by an instance of the neutral data format Industry Foundation Classes (IFC). With reference to the temporal and spatial data, the promoted IFC class structure is optimized for the particular query functionality. On the basis of this adapted object model we make use of the Language-Integrated Query technology (LINQ) provided by Microsoft's .NET framework. As LINQ was originally developed for defining static queries which are evaluated at compile time, we have implemented a dynamic version called Live LINQ in order to allow the user to flexibly formulate queries and to process these during the runtime of the BIM application. The developed interface provides a powerful and easy-to use mechanism for the spatial-temporal analysis and verification of 4D building information models.

BIM QUERIES

The concept of Building Information Modeling (BIM) is based on the use of intelligent, machine-readable representations of a building over its entire lifecycle. Using these representations, it is possible to provide extended computational functionality such as interactive query evaluation involving spatial and temporal predicates. This technology is well known from database management systems. However, BIM data models have particular characteristics that necessitate special implementations to realize the desired query support. In its standard form, a building model combines 3D geometry information with a complex object structure of components and their interrelations. By capturing construction schedule data, the BIM model can be extended to include temporal information concerning the construction and/or installation of individual components. This turns a conventional BIM model into a 4D model (Fischer, 2003).

Today, these 4D models have to be manually checked for correctness, which is a laborious and error-prone process. Based on our previous work concerning the realization of a spatial query language (Borrmann & Rank, 2009a,b), in this paper we discuss their extension through the inclusion of temporal operators. The resulting spatio-temporal query language facilitates the computational analysis and validation of 4D building models. An example of a spatio-temporal BIM query is "Select all non-supporting walls which touch floor 1 and ceiling 1 and which have been completed after ceiling 1". This request combines the examination of temporal, spatial and semantic predicates. Furthermore, the query is an example for the validation of a 4D model. The result represents all non-supporting walls in one particular storey. The system presented in this paper is able to process this query.

As input for the developed spatio-temporal query functionality, we make use of the vendor-neutral product model Industry Foundation Classes (IFC). However, the IFC object model has a complex and extensive data structure, particularly with respect to the geometric and temporal properties of building components, which makes the formulation of spatio-temporal queries extremely complicated. We therefore propose to transform this into a streamlined data structure. Combined with an extended version of the Language Integrated Query (LINQ) library, the system provides end users with an easy-to-use syntax for the formulation of queries involving spatial, temporal and semantic information.

RELATED WORK

One of the main functionalities of geographical information systems (GIS) is the processing of spatial queries. Though most of the available commercial GIS products restrict the spatial query functionality to 2D space, there has been ongoing, intensive research into 4D data storage and appropriate query technology (Noh, 2004). At the same time, the different modeling approaches in GIS and BIM make the direct application of GIS query technology for Building Information Modeling impossible. Entities in geographical information systems are set up as separate objects. They are often brought in a common context solely by their spatial location. BIM by contrast promotes a hierarchical product model with complex relations between the diverse components (buildingSMART, 2012). In addition, the representation of temporal data in GIS is not suitable for analyzing the construction and lifetime intervals of components in a BIM construction schedule.

Current research in BIM addresses methods for model filtering and the extraction of valid model views. General approaches using standardized database technology (SQL) have to be distinguished from domain-specific implementations. The complex IFC data structure and the associated object-relational mapping can lead to problems in performance and scaling when relational databases are employed (Wiet, 2012). Domain-specific developments range from native queries in the EXPRESS-X language to schema-based approaches such as GMSD (Weise, 2003) and PMQL, the Product Model Query Language of the EuroSTEP Model Server. Using these approaches, the creation of even a simple query can become very cumbersome (Borrmann & Rank, 2010). BIMQL is a promising research project realized as part of the bimserver.org platform (Wiet, 2012). This language operates

directly on the native IFC data format. The approach of using query shortcuts is also employed in our proposed system for geometry and temporal information.

DEFINITION OF THE TEMPORAL OPERATORS

The available data of the construction schedule represents the anticipated construction times of components. The user of the proposed BIM query system can additionally store information about the estimated lifetime of building parts. This makes it possible for the system to handle temporal queries concerning the lifetime interval of components.

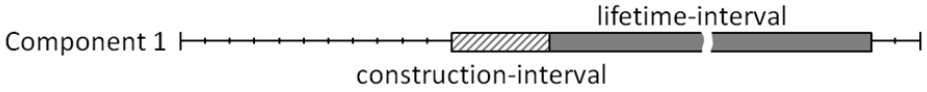

Figure 1: Construction interval and lifetime interval of component 1

To establish a common understanding of the meaning of the temporal operators, we institute a formal definition of their semantics based on the definitions provided by (Vilain, 1982). The temporal primitives employed here are intervals and points in time. The possible relations between these primitives are reflected by the 14 temporal predicates depicted in Table 1, which are implemented in our presented system. Using these predicates we are able to analyze the temporal relationships in a construction schedule.

	Temporal Operator		Temporal Operator
P_1 P_2	P_1 before P_2 P_2 after P_1	I_1 I_2	I_1 before I_2 I_2 after I_1
P_1 P_2	P_1 equals P_2 P_2 equals P_1	I_2 I_1	I_1 contains I_2 I_2 during I_1
P I	P before I I after P	I_2 I_1	I_1 begins I_2 I_2 begunBy I_1
P I	P begins I I begunBy P	I_1 I_2	I_1 ends I_2 I_2 endedBy I_1
I P	P during I I contains P	I_1 I_2	I_1 overlaps I_2 I_2 overlappedBy I_1
I P	P ends I I endedBy P	I_1 I_2	I_1 meets I_2 I_2 metBy I_1
I P	P after I I before P	I_1 I_2	I_1 equals I_2 I_2 equals I_1

Table 1: Temporal operators of the query language, based on (Vilain, 1982)

IFC DATA MODELING

The IFC standard includes about 600 classes, providing extensive data structures for the comprehensive and detailed modeling of buildings. IFC as a neutral data format has particular importance as it enables data exchange between software products from

different vendors, and serves as well as for realizing the hand-over of building data from the designers and contractors to the building operators.

The BIM query system we have developed uses ifcXML data conforming to the ifcXML2x3 standard as input (ifcXML2x3, 2012). A particular challenge is that the IFC make use of a very fine-grained object model involving a large set of inter-connected entities. In addition, the relationships between these entities are modeled as explicit objects. This makes the formulation of spatio-temporal queries using the original IFC schema extremely complicated. As a consequence, we propose adapting the model for the specific requirements of spatio-temporal analysis. This model resolves the objectified relationships and combines fine-grain data into more meaningful structures.

In the following, we describe our modifications of the IFC class structure to include temporal data. In the original model, scheduled construction intervals are represented by a branched structure composed of four different classes: The IfcRelAssignsToProduct class associates an IfcProduct, e.g. a wall, with a corresponding IfcTask. The task again refers to an IfcTaskTime, which finally includes a start and finish time (Figure 2).

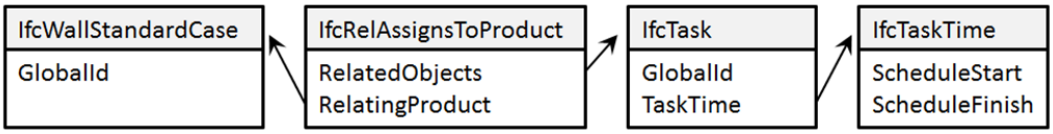

Figure 2: Relations between components and their construction times in the IFC

The data structure is simplified by adding a direct link between the IfcProduct object and the IfcTaskTime object. Thus, the recurring traversal of branches is not required for temporal queries. Furthermore, the relation arises at an IfcProduct which is a more natural starting point for use in a query than to examine a set of objectified relationships. The streamlined data structure is depicted in Figure 3.

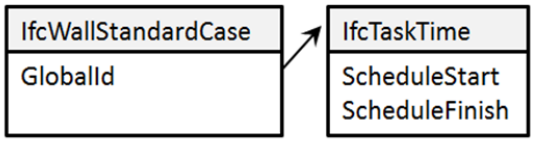

Figure 3: Simplified structure of components and their construction times

The geometrical information of IfcProduct objects is even more widely scattered across the IFC model. However, for the evaluation of the topological predicates explicit geometry is needed. Thus, the geometry information is transferred to explicit triangular meshes by use of external software. The resulting meshes are then linked to the appropriate IfcProduct objects.

SPATIAL AND TEMPORAL ALGORITHMS

The evaluation of the temporal algorithms is based on a comparison of the interval and points in time. For example, the start and end points are examined to compute

whether two intervals are overlapping. The speed of execution of these tests is fast enough to obviate the need for a supporting indexing structure.

However, the execution of topological operators is expensive and can slow down the system. To alleviate this a spatial index using an R* tree (Beckmann et al. 1990.) is constructed for IfcProduct objects. This eliminates many candidates using inexpensive pretests in topological queries. The topological algorithms are based on octrees. Here, the geometry of two IfcProduct objects is transferred on-the-fly to corresponding octrees. The cells of the tree, called octants, are classified by color attributes. The colors reflect the octant's spatial location concerning the hull of the original geometry. Possible color values are Black (Inside), Gray (Boundary) and White (Outside). By using a flooding approach, three colored octrees are produced. In the parallel traversal of two trees, color combinations are recognized. The 9 Intersection Model (9-IM) introduced by (Egenhofer, 1989) can now be used to deduce the accurate topological predicate. The collected color combinations of octants serve as input data. A detailed description of the implementation of topological operators is presented in (Daum & Borrmann, 2012).

FORMULATION OF FILTER EXPRESSIONS AND THEIR EVALUATION

In the proposed concept, a query expression is entered as source code by the end user or application programmer, respectively. The expression defines a predicate, which is used to select objects from a given set. The objects in the returned result set satisfy the predicate in the expression. By way of example, a set of walls is filtered by the expression `wall.GlobalID.StartsWith("1pJQ")` . Here, `wall` is a formal parameter which is replaced by all walls in the set one after another.

For the evaluation of query expressions at a set level, we use LINQ as it provides powerful query mechanisms for in-memory collections and object networks. LINQ is neatly integrated into the .NET framework and queries can be formulized by any .NET language. The queries are type safe and attributes and methods of involved objects can be used. For the definition of a query, an anonymous function, called a Lambda expression, like `wall => wall.GlobalID.StartsWith("1pJQ")` is used. The developed BIM system uses C# in combination with LINQ to examine in-memory collections. This means the filtering of building models can be executed directly in the fast main memory of the machine. When a 64-bit operating system is used, even very large building models can be queried without data having to be swapped to slower secondary storage. This is not possible with conventional database technology.

LIVE LINQ

When used as a standard component of .NET, LINQ requires the definition of filter predicates at compile time. While it is possible to pass parameters to a LINQ query at runtime, the system is restricted to pre-defined filters. To support flexible query statements that are entered by the end user at runtime, the expression must be converted to an executable filter object. This is achieved by on-the-fly compilation of user input at runtime. We call this functionality Live LINQ. If the entered C# code is accurate, it is compiled and saved as a .NET assembly, shown as a DLL in the file system. The query application can then load this assembly at runtime and use the

contained filter object in the pending filter execution. An unfiltered set of IfcProduct objects is supplied to the user as a data source. If a query is executed, the LINQ processor iterates over the IfcProduct collection. The returned set only contains entities, for which the dynamic filter evaluates to true.

As mentioned above, the IFC class structure can lead to complex filter statements. To make it easier to formulate filters, so-called query shortcuts are used and implemented with extension methods provided by the .NET framework. Extension methods make it possible to publish new methods on already existing types without having to recompile the original type. The approach of dynamic DLL creation and import is comparable to the query case: the difference is that the user enters a new extension method instead of a query. This method is compiled and loaded to the BIM application process. The extension method is automatically attached to the specific type. The user can simplify the query, as shown in Example 5, using a query shortcut.

In contrast to standard LINQ, the usage of Live LINQ allows the user to flexibly define queries. All object attributes can be used in a query and the system is not restricted to predefined filters. Additionally, as LINQ directly interact at an object level, methods published by objects are also accessible in queries, which make the temporal and topological operators available. Finally, filter generation at runtime means the compiler can be used as a validation tool for user-defined query statements. If inadequate code is submitted, the compiler returns meaningful error messages.

QUERY EXAMPLES

Example 1: Select walls with GlobalId 1pJQicIrH4UR_2BqvRP

```
IfcProducts.Where(p =>
        p is IfcWallStandardCase && p.GlobalId  == "1pJQicIrH4UR_2BqvRP")
```

Example 2: Select walls which overlap with wall 1 (index 25)

```
IfcProducts.Where(p => {
        var wall1 = IfcProducts [25];
        return p is IfcWallStandardCase  && p.Overlaps(wall1); })
```

Example 3: Select walls, if their construction-interval overlaps with that of wall1 and if they meet wall1

```
IfcProducts.Where(n => {
        var wall1 = IfcProducts [25];
        var constructionInterval1= wall1.ConstructionIntervall;
        return p is IfcWallStandardCase  &&
                p.TOverlaps(constructionInterval1) && p.Meets(wall1);})
```

Example 4: Select walls made of concrete

```
IfcProducts.Where(p =>
        p.hasAssociations[1].relatingMaterial.forLayerSet.
        materialLayers[1].material.name == "concrete")
```

Example 5: Select walls made of concrete (using the extension method)

IfcProducts.Where(p => p.GetMaterialName() == "concrete")

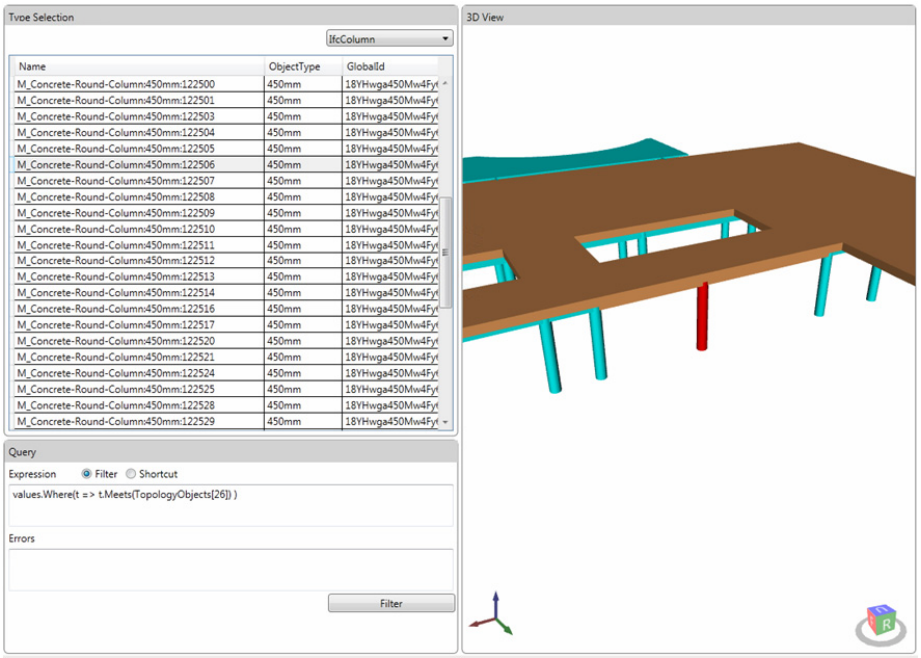

Figure 4: BIM system with spatio-temporal query functionality and LIVE LINQ

CONCLUSION

In this paper, we described the definition and implementation of a spatio-temporal query language for BIM. This language facilitates the computer-aided analysis of 4D models and the extraction of well-defined sub-models. As a basis for the query language we use the IFC standard, which provides a powerful modeling environment for the representation of buildings. Because the IFC class structure is only partially suitable for queries which directly involve spatial and temporal relationships, we propose a simplified data model adapted to meet the needs of representing and querying spatio-temporal properties and relationships. This significantly simplifies the formulation of query statements and reduces the computational costs of query evaluation. At the same time, the system is also capable of querying the original, complex IFC data structure. The queries are formulated as C# code and executed after an on-the-fly compilation in the LIVE LINQ environment. Object attributes and methods can therefore be included in query statements. In this way, spatio-temporal predicates are made available for direct query formulation. Using the described approach, we have been able to provide flexible BIM query functionality that provides spatial and temporal awareness.

REFERENCES

Beckmann, N., Kriegel, H.-P., Schneider, R., Seeger, B. (1990): The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. In: Proc. of the 1990 ACM SIGMOD International Conference on Management of Data.

Borrmann, A., and Rank, E. (2009a). "Topological analysis of 3D building models using a spatial query language" *Advanced Engineering Informatics* 23(4), 370-385.

Borrmann, A., and Rank, E. (2009b). "Specification and implementation of directional operators in a 3D spatial query language for building information models", *Advanced Engineering Informatics* 23 (1), 32-44

Borrmann, A., and Rank, E. (2010). "Query Support for BIMs using Semantic and Spatial Conditions" *Handbook of Research on Building Information Modeling and Construction Informatics: Concepts and Technologies,* IGI Global.

buildingSMART Ltd. (2012). "IFC data schemas" *Industry Foundation Classes* http://www.buildingsmart-tech.org/ifc/IFC2x3/TC1/html/index.htm> (10 January 2012).

Daum, S., and Borrmann, A. (2012). "Efficient and Robust Octree Generation for Implementing Topological Queries for Building Information Models" *Proceedings of the 19th EG-ICE International Workshop*, EG-ICE.

Egenhofer, M., Herring, J., (1989). "A mathematical framework for the definition of topological relationships." *Proc. of the 4th Int. Symp. on Spatial Data Handling.*

ifcXML2x3, (2012)."ifcXML reference", http://www.buildingsmart-tech.org/> (10 January 2012).

Fischer, M., Haymaker J., and Liston K. (2003) "Benefits of 3d and 4d Models for Facility Managers and AEC Service Providers", *4D CAD and Visualization in Construction,* Developments and Applications

Noh, S.-Y. (2004). "Literature Review on Temporal, Spatial, and Spatiotemporal Data Models." *Technical Report - Computer Science.* Iowa, USA, Iowa State University.

postGIS (2012). "Using PostGIS: Data Management and Queries", *postGIS reference* http://postgis.refractions.net/documentation/manual-2.0/using_postgis_dbmanagement.html#DE-9IM> (10 January 2012).

Vilain M. B. (1982): "A system for reasoning about time" AAAI-82 Proceedings

Weise, M., Katranuschkov, P., and Scherer, R. J. (2003). "Generalized model subset definition schema." *Proceedings of the 20th Conference on Information Technology in Construction.* CIB-W78.