Workshop: Developing Building Information Model Visualizations Using a Domain Specific Language

Helga Tauscher[1] , Raimar J. Scherer[2]
[1,2] TU Dresden, Faculty of Civil Engineering, Institute of Construction Informatics
[1,2] http://tu-dresden.de/bau/cib
[1,2] {helga.tauscher|raimar.scherer}@tu-dresden.de

The rise of digital building models has devalued the broad domain of architectural visualization, a former core topic of the domain. At the same time, digital media has opened up new possibilities for interactive and explorative visual representations. Against this background the workshop advocates to rediscover visualization as a distinct topic in the context of architecture and construction. The workshop introduces a method and a theoretic framework for the creation of visual representations from building information models under involvement of architects and engineers as domain experts, and a prototypical implementation, which serves as a proof of concept and allows for the practical application of the method. The workshop presents the prototype based on selected hands-on examples.

Keywords: *BIM, Visualization, DSL*

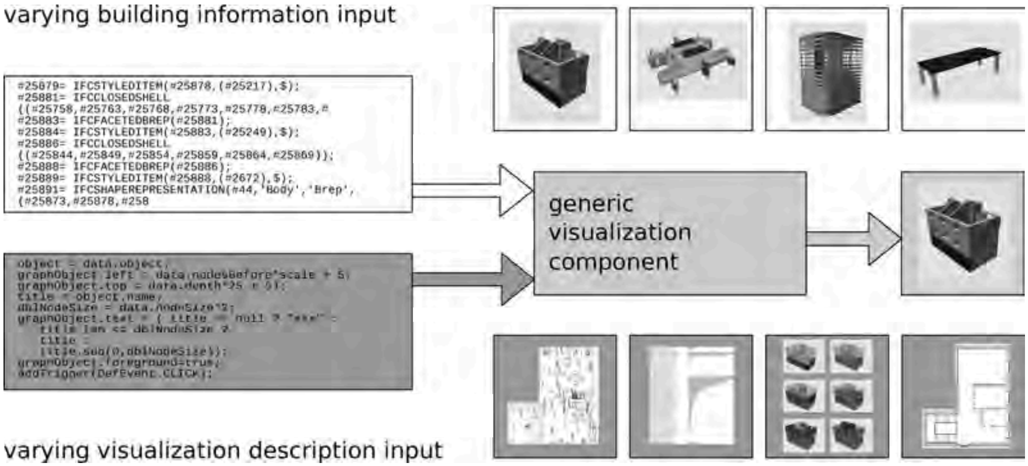
INTRODUCTION AND BACKGROUND

Visual representations and presentations have always been core architectural topics. Since the subject of architectural design is overly complex, visualizations are required for architects and engineers to communicate to themselves while designing and constructing as well as to other project participants. However, with the rise of digital building models, the task of creating visual representations has taken a back seat, and so have visualization skills. Nowadays, software tools create drawings, animations, renderings, and diagrams on behalf of the architects and engineers. The specification of visualization details is thus left to user interface designers and software engineers.

Digital media has at the same time opened up new possibilities for representation and spawned the discipline of information visualization, where research from graphics, psychology, human-computer interaction, and computer interaction converges into research on the general question how to visually represent abstract information. Building upon those developments time is ripe to rediscover visualization as a distinct topic in the context of architecture and construction.

METHOD AND THEORETIC FRAMEWORK

At the institute for construction information at TU Dresden we proposed a method and a theoretic framework for the creation of visual representations from building information models under involvement of domain experts. This method allows to specify visualizations using a domain specific lan-


varying building information input
generic visualization component
varying visualization description input

```
#25879= IFCSTYLEDITEM(#25878,(#25217),$);
#25881= IFCCLOSEDSHELL
((#25758,#25763,#25768,#25773,#25778,#25783,#
#25883= IFCFACETEDBREP(#25881);
#25884= IFCSTYLEDITEM(#25883,(#25249),$);
#25886= IFCCLOSEDSHELL
((#25844,#25849,#25854,#25859,#25864,#25869));
#25888= IFCFACETEDBREP(#25886);
#25889= IFCSTYLEDITEM(#25888,(#2672),$);
#25891= IFCSHAPEREPRESENTATION(#44,'Body','Brep',
(#25873,#25878,#258
```

```
object = data.object;
graphObject.left = data.nodesBefore*scale + 5;
graphObject.top = data.depth*25 + 5);
title = object.name;
dblNodeSize = data.nodeSize*2;
graphObject.text = ( title == null ? "xxx" :
    title.len <= dblNodeSize ?
    title :
    title.sub(0,dblNodeSize));
graphObject.foreground=true;
addTrigger(DefEvent.CLICK);
```

Figure 1
Visualization generation from varying building information models and varying visualization specifications

guage (DSL). A particular visualization specification is then processed together with a building information model to produce a particular visualization. This way a particular building information model can be represented in different ways, and a particular visualization specification can be reused for multiple building information models as shown in figure 1.

The suggested software architecture of a respective processing application follows the paradigm of the visualization pipeline (Haber 1979). The pipeline model breaks the visualization process into three successive steps: filter, map, and render. Filtering and mapping are general issues of information technologies, which have also been studied in the context of building information models (e.g. Katranuschkov 2000). Since previous work on building information model filtering and mapping exists, visualization generation can be seen as a specific application of these techniques, with a specific target space for the mapping - the space of potential visualizations.

The specifics of the mapping task, the visualization domain and the building information modelling domain allow to narrow down the general purpose mapping concepts. A DSL to describe visualizations has to cover the following concepts: Filters allow to query building information models and to select objects and attributes. These selected parts of the building model are then to be mapped onto visualization objects and their attributes. Simple visualization mappings must be combinable to form more complex mapping specifications.

PROTOTYP AND DSL IMPLEMENTATION

Prototypical implementations of both the DSL and the processing application serve as proof of concept and allow for the practical application of the methods. We called the DSL implementation Building Information Style Language (BISL), in reference to other style languages such as the extensible style language (XSL). The processing engine implementation is called Billie, the Building Information Style Engine.

Billie is implemented with a modular architecture according to the visualization pipeline paradigm as shown in figure 2. The modular architecture allows to extend and update single modules such as the visualization models or query languages and engines. Therefore the application acts more like a framework, integrating diverse libraries under a common interface. Since Billie is implemented in Java and runs in a JVM, the dynamic programming language Groovy is used in order to leverage its DSL features.

The prototype can be used as a library to be integrated in other applications or standalone as a commandline application. The parameters to be given to the engine consist of a building model to be visualized and a visualization specification. The visualization specification can either be given as a file written in the BISL DSL or as a precompiled Java archive which is then run dynamically as a service implementation (e.g. Knoernschild 2012).

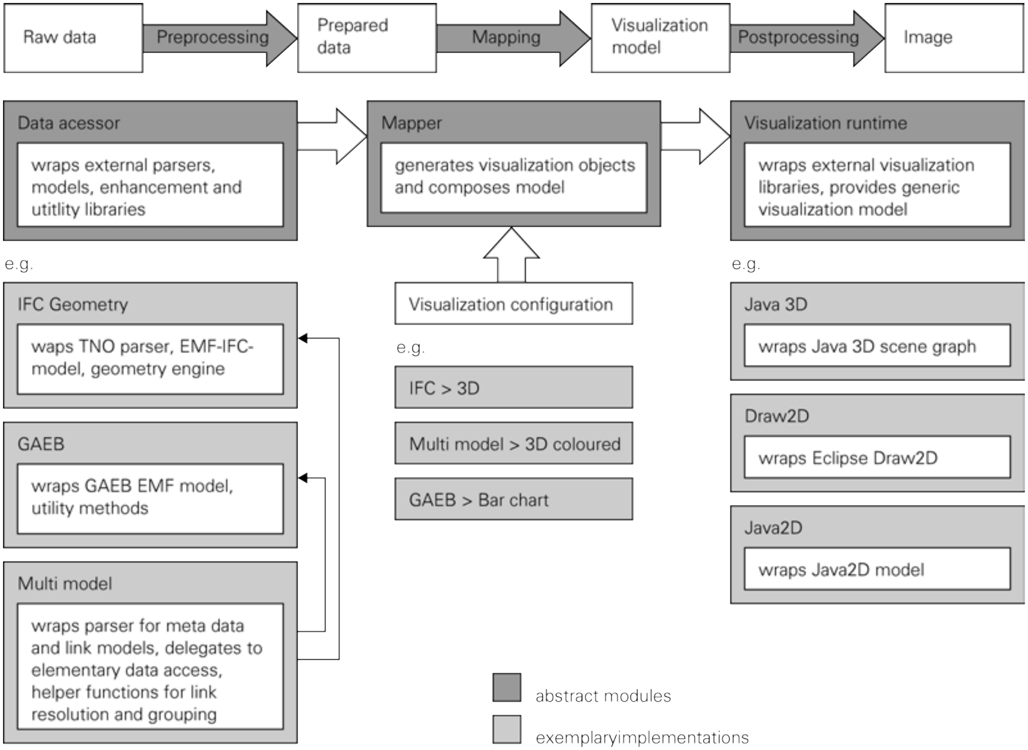
The DSL elements which are implemented include the most important visualization objects: rectangles, polygones, text, lines, bezier curves, boxes, polyeders. The list of implemented objects is not exhaustive yet. Apart from the spatial properties of objects, their shape and size, the DSL allows to specify temporal properties as changes in time. The combination of simpler visualization specifications to more complex ones may happen in both space or time.

Documentation of both the BISL and Billie can be found at the Github project page [1]. The documentation contains also sample cases with respective DSL code.

TOWARDS A CONSTRUCTION VISUALIZATION COMMUNITY

The workshop is intended to present the aforementioned work, but also to foster architectural and construction specific visualization as emerging fields and to constitute a respective community.

Thus, in the workshop, the prototype is first introduced based on given example visualizations and their descriptions. Participants then proceed to cus-



Raw data
Preprocessing
Prepared data
Mapping
Visualization model
Postprocessing
Image
Data acessor
wraps external parsers, models, enhancement and utitlity libraries
Mapper
generates visualization objects and composes model
Visualization runtime
wraps external visualization libraries, provides generic visualization model
e.g.
IFC Geometry
waps TNO parser, EMF-IFC-model, geometry engine
GAEB
wraps GAEB EMF model, utility methods
Multi model
wraps parser for meta data and link models, delegates to elementary data access, helper functions for link resolution and grouping
Visualization configuration
e.g.
IFC > 3D
Multi model > 3D coloured
GAEB > Bar chart
e.g.
Java 3D
wraps Java 3D scene graph
Draw2D
wraps Eclipse Draw2D
Java2D
wraps Java2D model
abstract modules
exemplaryimplementations

Figure 2
Modular architecture of the visualization processing engine

tomize given visualization descriptions and finally design a custom visualization use case from scratch. The visualization use cases serve as a foil for the introduction of technical details, limitations and possibilities for the extension of the framework. The material for the workshop is also used for an online tutorial accessible at the project website.

The examples for the application of the visualization framework produced in the workshop are also documented at the website. The outcome includes conceptual sketches, implementation concepts and partial realizations for visualization techniques developed by the participants. Depending on the mixture of participants, their skills, interest and prerequisites, an extension to the framework may also result from the workshop.

The workshop is rounded off with the presentation and discussion of possibilities to contribute to further development and extension of the framework, and to further research and discourse around building information visualization. To this end, the project website provides software downloads, issue tracking, as well as a wiki and discussion forum.

REFERENCES

Haber, RB and McNabb, DA 1979, 'Visualization Idioms: A Conceptual Model for Scientific Visualization Systems', in Nielson, GM, Shriver, B and Rosenblum, LJ (eds) 1979, *Visualization in Scientific Computing*, IEE Computer Science Press, Los Alamitos, CA

Katranuschkov, P 2000, *A Mapping Language for Concurrent Engineering Processes*, Ph.D. Thesis, TU-Dresden

Knoernschild, K 2012, *Java Application Architecture. Modularity Patterns with Examples Using OSGI*, Prentice Hall, Upper Saddle River, New Jersey

[1] http://hlg.github.io/billie