Journal of
open research software

Tauscher, H 2018 Billie. A Prototypical Framework for Building Information Model Visualization. *Journal of Open Research Software*, 6: 18. DOI: https://doi.org/10.5334/jors.206

SOFTWARE METAPAPER

Billie. A Prototypical Framework for Building Information Model Visualization

Helga Tauscher
National University of Singapore, SG
mail@helgatauscher.de

Current software applications for Building Information Modelling provide only limited possibilities to create customized visual representations of designed buildings and planned constructions. With reusable and exchangeable visualization configurations, customized domain- and task-specific visual representations could be generated for given building information. This concept is demonstrated with a prototypical implementation: Billie is a proof-of-concept framework written in Java that accepts building information together with a visualization specification and produces scene graphs for the customized visualization of the given building information.

To achieve configurability, Billie provides abstract classes as extension points to implement new BIM input data accessors, new configurations for generating the visualizations, and new scene graph types as visualization targets. With a domain specific language, the configurations become independent of their runtime environment and can be passed between different software applications. The software can be reused for research on new visualization methods in architecture, engineering, and construction or as a supporting visualization tool for research on other topics in the area.

Keywords: Configurable visualization; visualization framework; building information model
Funding statement: The development of the software was partially (during the early stages) supported financially in the course of the projects mefisto (grant no. 01LA09001) and eworkBau (grant no. 01PF07045A) by the German Federal Ministry of Education and Research, which is gratefully acknowledged.

(1) Overview

1.1 Introduction

In the architecture, engineering and construction (AEC) field, it has always been necessary to exchange information between the involved parties throughout the whole cycle of design, planning, construction, and operation of construction works such as buildings. To this end, drawings, tables, diagrams, and text documents describing the physical artefact to be constructed were produced and circulated among project stakeholders. In the digital era, with the rapid evolution of information technology, digital tools are naturally adapted and applied in AEC. In order to be processed by digital tools the required information needs to exist in digital form, and is thus supposed to be captured, exchanged and stored digitally. Models, standards, and software to achieve this goal were progressively developed during the previous years and are successively adapted in the practices of the AEC industry under the umbrella term *Building Information Modelling* (BIM) and mediated through open standards [1].

In the context of this paper, *building information models* are understood as "digital representation of the future building, its properties and its production process in a semantically explicit form" [2]. This definition distinguishes building information models from other digital representations, which contain the information only implicitly (scanned drawings or rendered imagery) or as explicit geometry with implicit semantics (2D or 3D CAD models). Traditional non-digital building information exchange and storage was based on drawings and documents, which are essentially visual representations. Nowadays, the exchange and persistence happens in digital, non-tangible, non-visual form – the visualizations are then generated on the fly in dedicated software tools. Thus, architects and engineers as the domain specialists have rarely the possibility to create or change those visualizations albeit the visualization of planned buildings was traditionally one of their core competences. However, visual representations and the active involvement in their creation are important means to access and understand the information. This is even more relevant, as the data captured in digital form is of continuously growing amount.

To reconstitute architects and engineers sovereignty over the visual representations of the information they produce, the author has, in previous work [3], proposed

the use of configurable visualizations which are described programmatically or with a dedicated domain specific language (DSL). A generic visualization component would then consume a visualization configuration together with respective building information to produce a specific visualization (**Figure 1**). These visualization configurations could be reused across projects and exchanged between parties involved in the planning process. They would allow for visualization solutions tailored to specific tasks and situations, for example to get a comprehensive overview of design changes after receiving an amendment for approval. The creation of a visualization configuration is meant to be carried out by domain specialists, which would thus be involved in the process of visualization generation and encouraged to experiment with visual representations and to re-explore their familiar field of AEC visualizations under the new paradigm of digital and interactive media. A use case study [4] identified four application areas, where such configurable visualizations could be particularly useful (communication, education, exploration, and experimentation) and provides examples for each area.

Although the idea of task-specific visual representations for building information has already been proposed in the early days of BIM, e.g. as so-called "monitors" [5], there exist few implementations that provide a flexible way to define such visualizations. Most commercial BIM software applicatons provide only fixed visualizations or arrangements of visualizations to switch between. A notable exception is RIB iTwo which allows to specify visualization properties such as colours through expressions that query into the building model [6, 7]. Most researchers that target the visualization of BIM models on a more general level recognize the need for task- and domain-specific visualizations, but still include only limited configurability in their solutions, e.g. [8, 9]. Others extend the configurability to real-time and promote adaptive visual representations that change after the visualization was generated [10].

There is a growing body of research work that studies the possibilities and limitations of computer visualization in transcending traditional paper-based workflows. By employing the computational power and the interactive capabilities of computers, these visualization approaches seek to not only represent the 3D geometry of a building better than in paper-based approaches, but also include key design parameters such as energy consumption or structural resilience, as well as the process of the building's construction and construction management related metrics. This way they hope to foster better understanding and discovery of spatio-temporal relations between objects and processes as well as critical points in the building design and in the construction schedule.

Most of these attempts stay close to the realm of experience by using visualization time to represent construction schedules in addition to the 3D geometry and enhance the resulting animation, e.g. with colours [11], additional graphical elements for zones [12], or as-built photographs [13]. Ivson et al. embark on a more experimental strategy with a 4D visualization where the 3rd dimension of the not-yet-built part of the building is partially sacrificed in favour of schedule information [14]. With the exception of the latter, where colour scales can be customized to reflect task-specific metrics, these prototypes do not consider configurations, but instead develop one specific visualization type that is optimized with regard to a concrete task and application scenario. Although this paper does not focus on the implementation of concrete novel visualizations, they can serve as use case examples guiding the development of a generic configurable solution in a bottom-up fashion. As such the author has also investigated concrete more experimental visualizations and their application in the AEC context, e.g. anamorphic maps [15], hierarchical

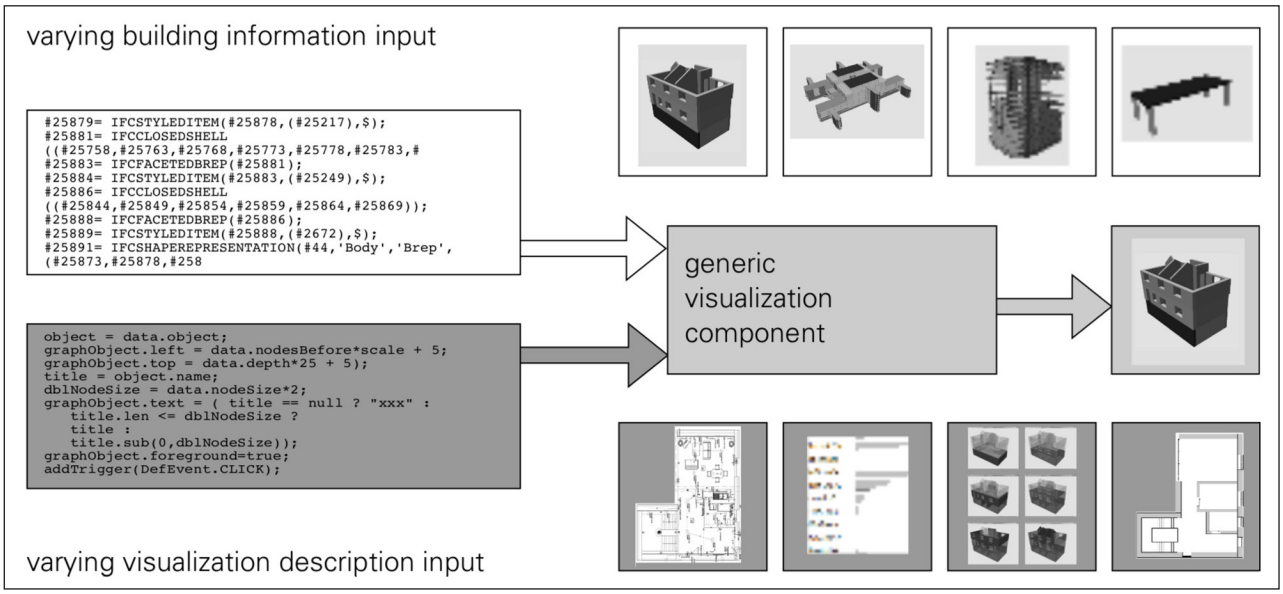

Figure 1: Generic visualization component configurable through visualization specifications.

edge bundles [16], and colour schemes for construction progress monitoring [17].

Another strand of recent development in the area of BIM visualization focusses on web viewers such as BIMsurfer [18] or WeXplorer, the visualization part of the xBIM project [19]. Since these developments are open source, they can be modified, used for the implementation of custom visualizations, or included into custom developments, as opposed to the aforementioned specialized visualizations, where neither source code nor binary libraries are publicly available. However, also these open source web viewers do not include runtime configurability natively and require substantial software development efforts to build custom visualization applications with them.

The approach taken with this work seeks to minimize required development efforts by providing all functionality except the specification of the visualization itself in a framework. This way developing a visualization is more like writing a configuration and the visualization specification constitutes additional input to the framework that is consumed just like the building information itself.

In order to verify the feasibility of the proposal, Billie was implemented as a proof-of-concept prototype. In parallel, use cases from the AEC sector have been studied and implemented as sample visualization configurations to complement the development of the prototype with example cases, including both more traditional as well as experimental use cases. Subsequently, Billie has been used to study the integration of the visualization process with other developments in construction informatics, such as the multimodel [20, 21] and BIM filter methods [22].

1.2 Implementation and architecture

The software prototype was implemented as a Java desktop application and library. It can be used through the standalone demo tool or be included into other applications. The next section describes the standalone usage before proceeding to the software architecture. For more information on integration with other applications please refer to Section 3.

1.2.1 Standalone usage

1.2.1.1 Installation

The standalone demo tool can be downloaded as a zip file (billie-0.12.zip) from Github https://github.com/hlg/billie/releases/tag/v0.12. Sample project data sets are provided with the release (billie-data.zip). Please refer to Section 1.3 for details about these data. To get started, download both the demo tool and the sample data zip files. Uncompress billie-0.12.zip into into a new directory and billie-data.zip into a subdirectory *data* inside. Open a command line and change into the new directory.

There are two ways to configure the visualization. For example, the visualization in **Figure 2** can be created either using the precompiled configuration named IFC_3D or the equivalent configuration expressed with a DSL in file ifc_3d-color.vis. In this section, the two possibilities are described for the preconfigured visualizations shipped with the release. The creation of custom visualizations for both approaches is described in Section 1.2.3. The command-line interfaces (CLI) for the two types of specifications are described in the next two sections.

1.2.1.2 Configuration runner

The configuration runner (de.tudresden.cib.vis.sampleApps.ConfigurationRunner) loads a selected precompiled visualization configuration *CONFIGNAME* and applies it to a building information model loaded from *BIMFILE*.

```
configurationrunner.bat [CONFIGNAME [BIMFILE]]
./configurationrunner.sh [CONFIGNAME [BIMFILE]]
```

If no *BIMFILE* is given, a file selection dialogue will prompt for the respective input file or folder. If no *CONFIGNAME* is given, all available configurations will be listed.

From the command prompt in the unzipped tool directory call either the batch file (*.bat, on Windows) or the shell file (*.sh, on Linux or MacOS). For example, the following command (on Linux or MacOS) will use the *IFC_3D* configuration and the IFC file from the carport project data set and subsequently produce the visualization shown in **Figure 2**.

```
./configrunner.sh IFC_3D data/carport.ifc
```

Hints on which configurations work together with which input files can be found in Section 1.3.

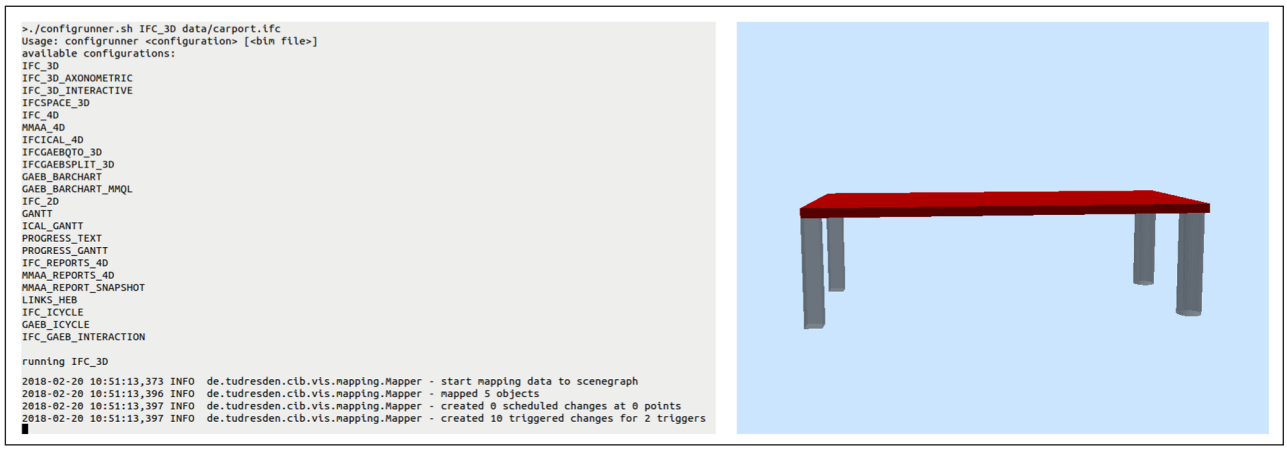

Figure 2: Visualization result from applying the IFC3D configuration to the ifc file from the carport sample data set.

1.2.1.3 DSL runner

The DSL runner (de.tudresden.cib.vis.DSL.VisDSLRunner) loads a visualization configuration from a DSL file *CONFIGNAME* and applies it to a building information model loaded from *BIMFILE*.

```
dslrunner.bat CONFIGFILE [BIMFILE]
./dslrunner.sh CONFIGFILE [BIMFILE]
```

If no *BIMFILE* is given, a file selection dialogue prompts for the respective input file or folder.

For example, the following command will use the *ifc_3d.vis* DSL specification and the IFC file from the carport project data set and produce the visualization shown in **Figure 2**.

```
./dslrunner.sh bisl/ifc_3d-color.vis data/carport.ifc
```

The DSL version of the visualization description is only implemented as a very rough sketch in the current alpha release and does not yet reflect the same functionality as the precompiled visualization description version.

1.2.2 Architecture

The prototype implements a modular architecture following the reference model of the visualization pipeline [23]. The visualization pipeline breaks the process of visualization generation into three successive steps: In the first step, relevant data is selected from the input and prepared for the following step. In the second step, the data is mapped to a visualization model. And finally in the third step, the image to be shown to the user is derived from this model. **Figure 3** depicts the modular architecture that results from employing this reference model for configurable visualizations.

Each and every pipeline step is covered by an abstract class serving as a plugin point for modules that implement the abstract class' interface and thus are exchangeable. This modular architecture realizes the adaptability of the system towards different requirements: Exchangeable data access modules allow for the adaptation to different types of input data. Exchangeable modules on the visualization side allow for the adaptation to different host environments for the visualization. Implementations of the central mapping step allow for different visualization configurations to guide the construction of the visualization. This central step plays a critical role because it is here where the configurability of the visualization is actually realized. Thus it is this part that is proposed to be implemented as a DSL, which then does not have to be precompiled, but will be parsed at runtime instead.

The generalizations on the two outer sides — the visualization and data side — are realized to different degrees, thus yielding different restrictions on the extent to which the modules are exchangeable independently

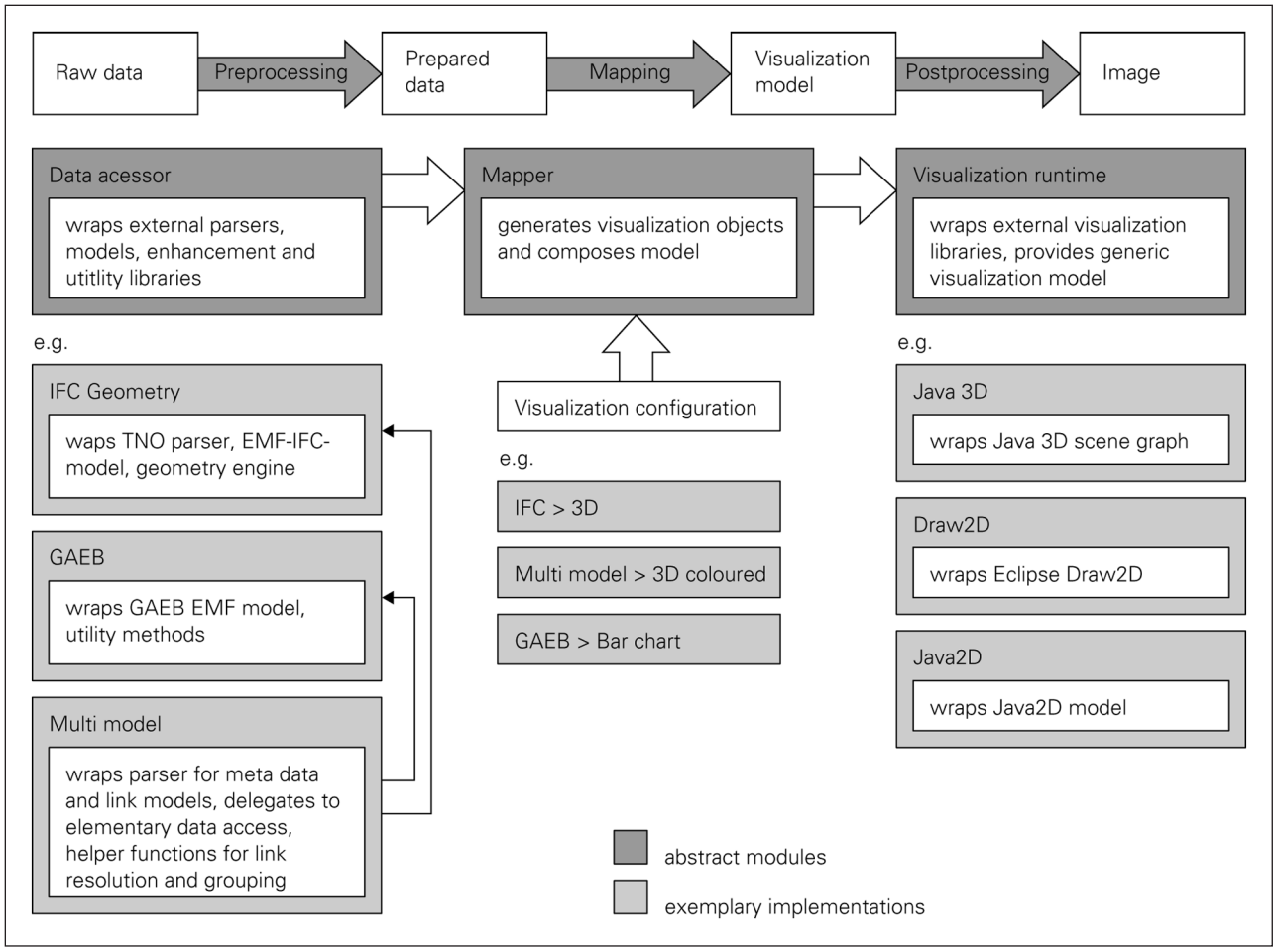

Figure 3: Generic visualization component modular architecture.

of the visualization configuration. On the visualization side, a high level of generalization was possible building on visualization ontologies [24], so that the modules can be exchanged transparently as long as the implementing visualization module provides the specific categories of functionality required for the given visualization configuration, such as 3-dimensionality or interaction. On the data access side however, the generalization is only less strict, since the advances and developments in the area of BIM are still ongoing and the range of possible input data is too heterogeneous to form simple categories. Thus there is a tighter dependency between the visualization configuration and the applicable data access module, compared to the dependency between the visualization configuration and the visualization module.

1.2.3 Custom Configurations

Both types of visualization configurations, precompiled and DSL specifications, can be used to realize custom task-specific visualizations. Complex visualizations consisting of smaller parts defined with self-contained configurations have been investigated [25], but have not been implemented in Billie.

1.2.3.1 Precompiled specification

For precompiled specifications, the abstract class de.tudresden.cib.vis.mapping.Configuration has to be extended. In particular, the custom configuration has to implement the *config* method. Inside the *config* method, the custom configuration can then call the designated configuration methods provided by the abstract class. The most important of these provided methods is *addMapping (Condition condition, PropertyMap mapping)*, which adds a mapping rule to be applied if a specific condition is fulfilled. The following example shows the implementation of the IFC_3D configuration used as an example in Section 1.2.1.

In order to process this configuration in conjunction with given input data (*ifcUrl*), the configuration has to be instantiated together with a matching data accessor and then be handed over to a mapper for the type of scene that should be produced, e.g. a Java3D scene:

```
Configuration ifc3dconfig = new Ifc_3D();
DataAccessor ifcAccessor = new EMFIfcGeometric
  Accessor();
ifcAccessor.read(ifcUrl);
Mapper ifcJava3dMapper = Java3dBuilder.createMapper
  (ifcAccessor)
Scene scene = ifcJava3dMapper.map(ifc3dConfig).get
  Scene();
```

To allow for a compile time compatibility check of the associated data accessor, mapper, scene graph, and configuration, the respective abstract classes have generified class and method signatures. However, due to type erasure in Java, generics will fail in the case of configurations specified by a DSL since these are only loaded at runtime. Thus the generic class signature might be replaced by some other mechanism in the future.

The Billie release includes a range of data accessors to consume the provided sample data as well as scene builders to generate respective scenes. These are listed below.

Data accessors:

- simple link-centred multimodel access
- grouped multimodel access
- GAEB access using the Eclipse Modelling Framework [26] with or without hierarchy analysis
- EMF-based IFC access with or without external geometry analysis and with or without external hierarchy analysis
- EMF-based access to quantity take-offs
- EMF-based access to schedules
- iCalendar schedules

Scene builders:

- Draw2D: creates Eclipse Draw2D models
- Java2D: creates Java2D scenes
- Java3D: creates Java3D scenes
- Text: creates text output, for debugging

```
public class Ifc_3D extends Configuration<EMFIfcParser.EngineEObject, Condition<EMFIfcPar.EngineEObject>>
   public void config() {
      this.addMapping(new Condition<EMFIfcParser.EngineEObject>() {
         @Override
         public boolean matches(EMFIfcParser.EngineEObject data) {
            return data.getObject() instanceof IfcBuildingElement;
         }
      }, new PropertyMap<EMFIfcParser.EngineEObject, VisFactory3D.Polyeder>() {
         @Override
         protected void configure() {
            Geometry geometry = data.getGeometry();
            if(data.getObject() instanceof IfcSlab || data.getObject() instanceof IfcRoof){
              graphObject.setColor(200,0,0,0);
            } else {
              graphObject.setColor(128,128,128,150);
            }
            graphObject.setVertizes(geometry.vertizes);
            graphObject.setNormals(geometry.normals);
            graphObject.setIndizes(geometry.indizes);
         }
      });
   }
}
```

The generated scenes are intended for further usage in custom applications. For convenience, the Billie release provides simple viewers for Java3D and Draw2D, offering a quick way to display the generated scenes. In addition, a configurable Java3D loader is provided as well.

1.2.3.2 Building Information Style Language (BISL)

The Building information style language (BISL) is proposed as a domain specific language (DSL) to specify visualization configurations at runtime. A basic mapping rule in BISL is specified by giving the type of the source and target object, a condition and an initial mapping as follows:

```
vt.rule(EngineEObject, Polyeder){
   condition {
      data.object instanceof IfcBuildingElement
   }
   initial {
      visual.vertices = data.geometry.vertices
      visual.normals = data.geometry.normals
      visual.indices = data.geometry.indices
      if (data.object instanceof IfcSlab)
         visual.color = [200,0,0,0]
      else
         visual.color = [128,128,128,150]
   }
}
```

The example specifies the same configuration as the precompiled example given in Section 1.2.3.1. By comparing the two listings, it can be seen that the DSL is merely a thin layer over the underlying domain model of the mapping rule specification as suggested by Fowler [27]. Details about the proposed DSL for visualization configurations including the main elements in Backus-Naur form (BNF) can be found in the authors thesis, Chapter 5.5 [2].

Only very basic mapping with simple animation and without interaction is implemented so far for the DSL interpreter. The Billie release contains four sample configurations (*.vis files).

1.3 Quality control

The software was tested with sporadic unit and integration tests which do not provide comprehensive code coverage, but are included in the Ant build scripts and thus executed on every build. Functional testing was done based on use case data described in the following.

A range of use cases from the AEC industry was collected before and in parallel to the development of the prototype. In the spirit of test-driven development, which promotes the implementation of unit tests before any actual implementation, these use cases served not only as after-the-fact functional tests, but also to guide the development of the prototypical implementation from the very beginning. Each use case consists of a sample data set from a building project and a visualization configuration, both of which in combination determine the resulting visualization. Both the building project data sets and the visualization configurations were designed to cover a broad and diverse range. Building project data sets (D3, D4, D5, D6) vary from small, artificially constructed demo projects to large buildings taken from real world projects. Visualization configurations (V2, V4, V5, V6) range from conventional visualizations, such as coloured 3D representations, bar charts, and Gantt charts to experimental visualizations, such as anamorphic maps and hierarchical edge bundles.

D3) **Carport:** four columns and one slab
D4) **One-family house:** full construction, multimodel with cost and schedule, implicit quantity-takeoff
D5) **High-rise building:** structural, core, shell, multimodel with cost, schedule, progress, quantity-takeoff
D6) **Airport building:** structural, core, shell, multimodel with cost, schedule, progress, quantity-takeoff

V2) **Elementary 3D models, bar charts, Gantt charts**
V4) **Simple multimodel visualizations**
V5) **Link visualization with HEBs**
V6) **Progress control visualization**

In **Table 1**, all those cases which can be reproduced with the current Billie prototype are listed in an adjacency matrix of input data sets and visualization configurations. The full list of considered cases contains further project data sets and configurations which can not yet be used within Billie [2].

To systematically test the implemented use cases, a set of sample project data is provided from the project site http://hlg.github.io/billie, covering the cases D3, D4, D5, and D6 listed above. Visualization configurations are shipped as precompiled configurations with the release. The script (*functional.sh* for Linux, *functional.bat* for Windows) contained in the Billie release lists all implemented pairs of these building project data sets and visualization configurations. It can be used to copy the command for a single combination and run this particular combination as a functional test. To run the functional test script semi-automatically with all combinations, the sample data is supposed to be unzipped into a *data* folder relative to the script. When running the test suite this way, each visualization is produced and displayed for examination and has to be closed manually before the script proceeds to the next case.

The testing was carried out under Ubuntu 12.04 LTS and Windows 7. No further measurement of memory and processor load has been done.

Table 1: Use cases as a combination of input data (rows) and visualization configuration (columns).

	V2	V4	V5	V6
D3	x	x	x	–
D4	x	x	x	x
D5	x	x	–	–
D6	x	x	x	x

(2) Availability

2.1 Operating system

Since Billie runs in the Java Virtual Machine, it is operation system independent, though the Java Runtime Environment needs to be installed. Testing was carried out on Windows 7 and Ubuntu 12.04 LTS. Native libraries are included in the release for 32- and 64-bit Windows, Linux, and MacOS system.

2.2 Programming language

Billie is implemented in Java 7 and Groovy.

2.3 Additional system requirements

Memory, disk and processor requirements depend on the size of the input data, which can be demanding for real-world construction projects with detailed geometry. The sample configurations are optimized for a screen width in the range of around 1000 pixels. For interactive visualizations a pointing device (e.g. a mouse) is required.

2.4 Dependencies

In the release, all dependencies needed for running the test cases are included. They are also sufficient to create custom configurations with the building information input types and scene graph target types listed in Section 1.2.3. Most dependencies are encapsulated in exchangeable modules. Dependencies are managed with Apache Ivy, the *ivy.xml* file in each module provides a detailed list of dependencies. The reminder of this section provides an excerpt from these files as compact overview.

Core modules:

- *vis*: Guava
- *vis.DSL*: Groovy

Data modules:

- *vis.data.bimserver*: Opensource BIMserver, EMF Ecore
- *vis.data.jsdai*: CIB BIMfit
- *vis.data.mmqlserver*: M2A2 MMQL
- *vis.data.multimodel*: CIB Multimodel, Mefisto GAEB/QTO/risk/schedule, EMF, ICal4J, Opensource BIMserver, JUnit

Visualization modules:

- *vis.runtime.draw2d*, *vis.scene.draw2d*: Eclipse Draw2D
- *vis.runtime.java3d*, *vis.scene.java3d*: Java3D

Sample applications:

- *vis.sampleApps*: Eclipse SWT
- *vis.swingApp*: Groovy Swing

Throughout the modules, also Apache Commons.IO, SLF4J, and JUnit are used.

2.5 List of contributors

Billie was created by Helga Tauscher as a research assistant at TU Dresden.

2.6 Software location

2.6.1 Archive

Name: Zenodo
Identifier: https://doi.org/10.5281/zenodo.1059067
Licence: GPL v2
Publisher: Helga Tauscher
Version published: v0.12
Date published: 28/04/18

2.6.2 Code repository

Name: Github
Identifier: http://github.com/hlg/billie
Date published: 19/02/18 (alpha v0.12)

2.7 Language

Java and Groovy

(3) Reuse potential

The software could be reused for research and development of use case specific construction visualizations from two possible perspectives. On one hand, as a prototyping framework it can facilitate research about the development and application of new interactive visualization methods in the field of AEC. For example it could be used to test the application of visual analytics methods in construction engineering. This approach seems promising to manage huge amounts of building data and keep it consistent during project time. Keim et al. [28] provide a research agenda for this emerging field. This type of research would focus on the visualizations themselves and strive to develop novel forms of visual representation.

On the other hand, it can aid other research on topics from the field of construction informatics that are not inherently visual, by providing a means to create visualizations for the research subjects to make them more tangible. Research approaches often introduce new conceptual models of the subject, which may imply a new domain view of the data or even information models with enhanced data. Naturally, for such novel domain models there might not yet be adequate support in existing software including visual representation. Billie could then be integrated in other research prototypes as a visualization component. Examples for this kind of non-visual research topics include e.g. computational fluid dynamics, risk analysis, and other simulations, or information management issues such as multimodels and filters.

The technical preconditions for reuse are given through the modular architecture. The creation of customized visualizations is described in Section 1.2.3. Using Billie simplifies the development process as opposed to from-scratch development, because existing data access and scene building modules (listed in Section 1.2.3) can be reused as a framework that handles and hides some aspects of the underlying libraries and thus facilitates rapid visualization prototyping. In addition, the included visualization runtime environments can be used for the quick instantiation of default viewer components.

In case of new requirements in terms of the input data or the application runtime context, new data accessors and scene builders can be implemented and plugged into the system. To achieve this, the following classes would have to be extended or implemented:

- de.tudresden.cib.vis.data.DataAccessor
- de.tudresden.cib.vis.scene.VisBuilder
- de.tudresden.cib.vis.scene.VisFactory2D/VisFactory3D
- de.tudresden.cib.vis.scene.UIContext

The modules prefixed with vis.data (for data accessors) and vis.scene (for scene builders) contain existing implementations or extensions and can serve as examples or starting points for custom developments.

Acknowledgements

I would like to acknowledge the support of Raimar J. Scherer, who provided the research environment to develop this software and acted as a mentor for the theoretical framework. I would also like to acknowledge the input and feedback I received from Sebastian Fuchs and Alexander Wülfing on the integration of Billie with the multimodel framework M2A2 and the filter library BIMfit.

Competing Interests

The author has no competing interests to declare.

References

1. **buildingSMART** 2018 Technical vision: About BIM [Internet]. Available from: https://www.buildingsmart.org/standards/technical-vision.
2. **Tauscher, H** 2017 Configurable nD-visualization for complex building information models [PhD thesis]. TU Dresden.
3. **Tauscher, H** and **Scherer, R J** 2016 Billie: An extendible framework for the configurable visualization of information models. In: Emmitt, S and Adeyeye, K (eds.), *Proceedings of the ID@50 Integrated Design Conference*, 111–24. Bath, UK.
4. **Tauscher, H** and **Scherer, R J** 2014 Use cases for configurable building information model visualization. In: *Fusion (ECAADE 32)*, 485–94. Newcastle, UK.
5. **Liebich, T** 1993 Wissensbasierter Architekturentwurf: Von den Modellen des Entwurfs zu einer intelligenten Computerunterstützung [PhD thesis]. Bauhausuniversität Weimar.
6. **Demharter, J** and **Scherer, R J** 2014 Visuelle Prüfung und Vergleich von Multimodellen. In: Scherer, R J and Schapke, S-E (eds.), *Informationssysteme im Bauwesen 1: Modelle, Methoden und Prozesse* [Internet], 483–502. Berlin: Springer. DOI: https://doi.org/10.1007/978-3-642-40883-0_20
7. **RIB Software SE** 2017 iTwo 4.0: Cloud-based 5D BIM enterprise solution [Internet]. Available from: http://www.itwo.com/en/5d-bim-enterprise-platform-itwo-4-0.
8. **Boton, C, Kubicki, S** and **Halin, G** 2013 Designing adapted visualization for collaborative 4D applications. *Automation in Construction*, 36: 152–67. DOI: https://doi.org/10.1016/j.autcon.2013.09.003
9. **Kuo, C-H, Tsai, M-H** and **Kang, S-C** 2011 A framework of information visualization for multi-system construction. *Automation in Construction*, 20: 247–62. DOI: https://doi.org/10.1016/j.autcon.2010.10.003
10. **Altenburger, T, Guerriero, A, Vagner, A** and **Martin, B** 2010 Toward adaptive context-aware user interfaces for better usability and productivity in AEC collaborative tasks. In: *Proceedings of the CIB W78 2010: 27th international conference*. Cairo, Egypt.
11. **Chang, H-S, Chih-Chung, K** and **Po-Han, C** 2009 Systematic procedure of determining an ideal color scheme on 4D models. *Advanced Engineering Informatics*, 23: 463–73. DOI: https://doi.org/10.1016/j.aei.2009.05.002
12. **Haque, M E** and **Rahman, M** 2009 Time-space-activity conflict detection using 4D visualization in multi-storied construction project. In: *Visual informatics: Bridging research and practice, first International Visual Informatics Conference (IVIC)*, 266–78. Kuala Lumpur, Malaysia.
13. **Golparvar-Fard, M, Peña-Mora, F, Arboleda, C A** and **Lee, S** 2009 Visualization of construction progress monitoring with 4D simulation model overlaid on time-lapsed photographs. *Journal of Computing in Civil Engineering*, 23: 391–404. DOI: https://doi.org/10.1061/(ASCE)0887-3801(2009)23:6(391)
14. **Ivson, P, Nascimento, D, Celes, W** and **Barbosa, S D** 2018 CasCADe: A novel 4D visualization system for virtual construction planning. *IEEE Transactions on Visualization & Computer Graphics*, 24: 687–97. DOI: https://doi.org/10.1109/TVCG.2017.2745105
15. **Tauscher, H** and **Scherer, R J** 2011 Area cartograms in building product model visualization: A case study on the presentation of non-spatial object properties in spatial context with anamorphic maps. In: *Respecting fragile places (ECAADE 29)*, 444–50. Ljubljana, Slovenia.
16. **Tauscher, H** and **Scherer, R J** 2013 Utilizing hierarchical edge bundles in multi model visualization. In: Suter, G, Wilde, P and de Rafiq, Y (eds.), *EG-ICE 2013: 20th International Workshop: Intelligent Computing in Engineering*, 351–60. Vienna University of Technology.
17. **Tauscher, H** and **Scherer, R J** 2013 Applied visualization methods for building information models with heterogeneous sources. In: *Envisioning Architecture: Design, Evaluation, Communication Conference (EAEA11)*, 93–100. Milan, Italy.
18. **bimsurfer.org** 2011 BIMsurfer: Open Source WebGL viewer for IFC models [Internet]. Available from: http://bimsurfer.org.
19. **Steve Lockley, C B** and **Černý, M.** Xbim.Essentials: A library for interoperable building information applications. *The Journal of Open Source Software*, 2: 473.
20. **Fuchs, S** and **Scherer, R J** 2017 Multimodels: Instant nD-modeling using original data. *Automation in Construction*, 75: 22–32. DOI: https://doi.org/10.1016/j.autcon.2016.11.013

21. **Tauscher, H** and **Scherer, R J** 2016 Divide and conquer, mix and match. a top–down and bottom–up approach to building information visualization. In: *Complexity & simplicity (ECAADE 34)*, 611–20. Oulu, Finland.
22. **Wülfing, A, Windisch, R** and **Scherer, R J** 2014 A visual BIM query language. In: *Proc 10th European Conference On Product And Process Modelling (ECPPM)*, 157–64. Vienna, Austria. DOI: https://doi.org/10.1201/b17396-30
23. **Haber, R B** and **McNabb, D A** 1979 Visualization idioms: A conceptual model for scientific visualization systems. In: Nielson, G M, Shriver, B and Rosenblum, L J (eds.), *Visualization in scientific computing*. Los Alamitos, CA: IEEE Computer Science Press.
24. **Voigt, M** and **Polowinski, J** 2011 Towards a unifying visualization ontology [Internet]. TU Dresden. Available from: http://nbn-resolving.de/urn:nbn:de:bsz:14-qucosa-67559.
25. **Tauscher, H** and **Scherer, R J** 2015 Specification of complex visualization configurations using hierarchically nested mapping rule sets. *ITcon* [Internet], 20: 40–50. Available from: http://www.itcon.org/2015/3.
26. **The Eclipse Foundation** 2018 Eclipse Modelling Framework (EMF) [Internet]. Available from: http://www.eclipse.org/modeling/emf.
27. **Fowler, M** 2010 Domain-specific languages. Boston: Addison-Wesley.
28. **Keim, D, Kohlhammer, J, Ellis, G** and **Mansmann, F** (eds.) 2010 Mastering the Information Age: Solving Problems with Visual Analytics. Goslar: Eurographics Association.

How to cite this article: Tauscher, H 2018 Billie. A Prototypical Framework for Building Information Model Visualization. *Journal of Open Research Software,* 6: 18. DOI: https://doi.org/10.5334/jors.206

Submitted: 26 November 2017 **Accepted:** 29 April 2018 **Published:** 18 May 2018

]u[*Journal of Open Research Software* is a peer-reviewed open access journal published by Ubiquity Press

OPEN ACCESS