A framework for multi-model collaboration and visualisation

S. Fuchs, P. Katranuschkov & R. J. Scherer
Institute of Construction Informatics, Dresden University of Technology, Germany

ABSTRACT: Information in construction processes is represented by various independent information models. Hence, the management of the resultant multi-model space is a very important issue for large scale construction processes. However, automated multi-model interoperability is still poorly supported by ICT. In this paper, a new framework for the integration of model collaboration and visualisation tools is proposed, as enabler for further research and development work in the area of model-based construction information management. The framework is implemented as a lean set of Eclipse plug-ins, each providing its own functionality. These plug-ins may have interdependencies to allow clients to reuse them. Consequently plug-ins can be information consumers and providers at the same time. The framework offers an extensible data model for a multi-model project, whereby plug-ins can take part in the global load-edit-save cycle.

1 INTRODUCTION

The large German BMBF research project Mefisto inaugurated in spring 2009 with duration of three years aims at overcoming client-contractor interoperability problems in construction processes based on partnership (Scherer et al. 2010). One of the project's main challenges is the management of the *distributed multi-model-space* defined via model schemas for building information, construction site and equipment information, costs, specifications, time schedules, organisation, processes, risks and uncertainties. The instances of these model schemas, i.e. the actual model data, need to be transformed, exchanged and managed horizontally (between client and contractor and among various discipline-specific representations), longitudinally (in their temporal development along the project phases) and vertically (on different levels of abstraction, to support different levels of decision making). An overview of these model assignments can be found in (Schapke et al. 2010).

The AEC community has developed IFC as a powerful extensible building information model which is increasingly gaining practical importance (Kiviniemi et al. 2008). However IFC does not (and is not intended to) store and carry all relevant data for all multi-faceted construction processes. Hence, nD modelling problems occur that are inherent to construction (cf. Aouad et al. 2007). There are approaches to extend IFC in order to cover more building process related information, as shown e.g. by Froese (2003), but such solutions have various limitations and assume IFC-conforming tools in all construction subdomains and subtasks, which is not the case in current design, construction planning and management software.

Therefore, based on the fact that not all relevant data can be structured in a single super schema, our approach is to take existing models as they are and treat them as one *interoperable multi-model space*. Advantages of that approach are as follows:

1 Existing and accepted data models like IFC or the German GAEB specifications model can be used further without modification;
2 According to a given task or process, information can be assembled in a straightforward way by composing relevant model data;
3 IT coverage of building process information can be extended by alternative data models, as suggested e.g. by Keilholz et al. (2009), or by data models created in future.

This shifts the paradigm from BIM-centred information management to a *federation of coequal multi-models* (Figure 1). The challenge is to give information access to a consumer (a person or software) as it would be by a single schema. This implies automated collaboration of the participating single models which is still poorly supported by current IT.

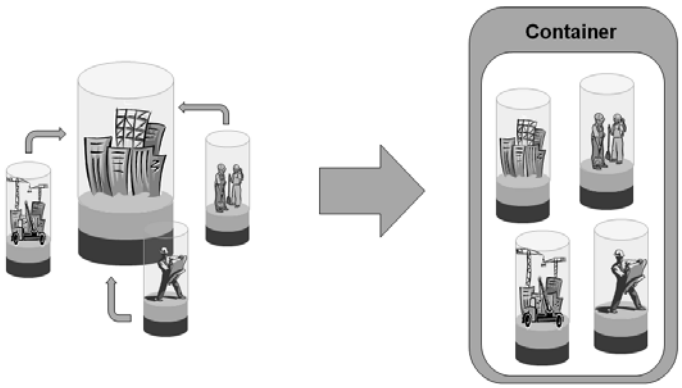

Figure 1. Paradigm shift from a BIM-centred approach (left) to a federation of coequal multi-models (right)

In this paper we present the main concepts and features of a novel framework for multi-model collaboration and visualisation, developed with the goal

to facilitate further research and development of multi-model management tools such as:
- Multi-model filtering
- Multi-model checking
- Multi-model mapping
- BIM-based and topological (network-like) multi-model visualisation
- Collision detection
- Process simulation.

2 THE MULTI-MODEL CHALLENGE

Consistent information access to the multi-model space is the key for general multi-model acceptance. In this chapter, the issues which drive the achievement of this goal are discussed.

Filtering

Current model filters should be extended to multi-model filters to allow combining selection criteria from the separate models. In analogy to geometry, domain specific collision detection can validate multi-model data, e.g. time checks, cost limits or personal responsibilities and any combination of them. A special aspect is versioning where two single model instances of the same real world object represent different states in the multi-model space. Compare methods can help to find and reproduce the changes between such states. This concept can even be extended up to versioning of multi-model instances.

Visualisation

Visualisation is used to present information in a context relevant way, appropriate for the actual supported process. Here the focus of work is on creating corresponding information views in relevance to very different visualisation technologies. This is in close relationship with the next addressed issue.

Data access

Since wee assume no common schema, neither explicit (as a formalised model) nor implicit (as a query interface), various basic questions arise:

1 What is the representation of a multi-model and how are the single-models accessed?
2 How are associations between the single-models established?
3 How can the combined information be accessed in a generic way?

These questions are subject of ongoing research, but there are just concepts and no comprehensive answers as yet.

Representation

In principle, the approach for multi-model information representation using a federation of coequal models can be based on so called *information containers*. Such containers can hold the single model data either by a URI reference to the physical document or data repository, or as embedded data. Given a XML serialisation, a corresponding link-element, respectively a CDATA section, can be used. Besides, some metadata must be carried for superior information process data and quality information about the single models. This metadata is not intended to provide multi-model collaboration by itself but can support that process by its context data.

Associations between the participating single-models are a key aspect of multi-model collaboration. Ideally, every real world object representation in a multi-model must be allocated to its different single model representations. Therefore semantically equivalent concepts in a model pair must be identified and have to be connected. While manual processes for that purpose are already established e.g. in standard cost calculation software, development of automatic *matching strategies* are one of the significant challenges in multi-model research. Matching may also imply various data transformations (mappings) that need to be taken into account.

Query language

To enable generic access of multi-model data, a common query language has to be defined. However, without the availability of a common schema this is very difficult to realise. Knowledge about structure and attributes of each single model and about their associations is necessary. A basic approach here can be to shift the responsibility for semantic concept matching to the query creator, in similarity to SQL where relation joins have to be defined at query level. Nevertheless, in order to avoid redundant manual or semi-automated matching strategies at query runtime, the association process should be performed before that point. Therefore, once made, associations between single models have to be kept accessible so that they can be quickly invoked on query execution. Consequently, such associations should have a serialisation. Thus, they can be embedded in the multi-model container to ensure multi-model consistency.

3 PROPOSED FRAMEWORK ARCHITECTURE

In this chapter we present the details of a proposed new software framework that helps for the solution of the outlined multi-model challenge.

To begin with, the roles in the framework lifecycle have to be defined. Accordingly, a person who develops the framework will be called *framework developer*, a person who uses the framework to develop a software product will be called *framework user* or *client*, and a person who uses the resulting product will be called *end user*.

The presented framework is intended to support the work of client developers and researchers on multi-model issues. Therefore its main *objectives* are to offer:

1 An extension mechanism to contribute the client software modules;
2 Modularity by itself and for the client modules to allow subset distributions;
3 A common user interface;
4 A possibility to load, hold and provide data of the single-model instances;
5 An implementation of the multi-model container format for persistence and data exchange.

Figure 2 provides an illustrative view of the overall concept, emphasising the idea of inter-connecting plugged-in clients and models in coherent manner with the help of the framework.

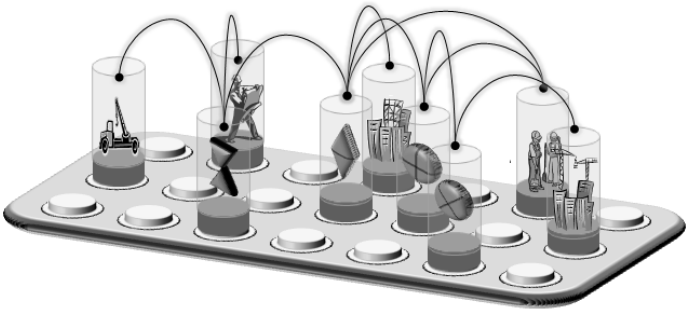

Figure 2. Overall concept of the proposed multi-model framework enabling development of interoperable plug-in clients

To comply with objectives 1 to 3, the Eclipse Rich Client Platform was chosen as runtime environment. Eclipse is built on Equinox, which is an open source implementation of the OSGi standard. As such it provides dynamic runtime modularity on top of a Java virtual machine (JVM). Software components, called bundles, can be run, stopped and updated without halting the JVM. In that way it is possible to deliver the framework, its libraries and a yet unknown amount of client components in a straightforward manner. Bundles describe their interdependencies in a manifest file, which allows a fine grained composing of runtime artefacts – as it is necessary for building subset distributions of the framework and client modules by framework users (D'Anjou et al. 2006).

Eclipse follows a *plug-in concept* by providing a generic extension mechanism via XML-descriptors. Plug-ins may offer extensibility by extension points and can contribute additional functionality by registration to other extension points. Due to the use of XML-descriptors, Eclipse can defer plug-in activation to the point where Java classes have to be loaded from it. This so called lazy loading allows a fast startup sequence and a small memory footprint, which is necessary when working with huge data models like IFC. Therefore, framework components and client modules are implemented as Eclipse plug-ins.

Furthermore Eclipse comes with the Standard Widget Toolkit (SWT), which is a Java library for the use of the operation system's native user interface elements. This enables a standard Look and Feel to the end user. An overview how to use Eclipse as an application platform can be found in (Gamma & Beck 2003).

Figure 3 below shows a screenshot of the graphical user interface of the developed multi-model framework, with an example set of IFC related plug-ins. The top left editor part is dedicated to the functional representation of each client plug-in, e.g. loaded single-model instances. Further parts can be contributed by every plug-in, like the IFC 3D viewer which is shown in the bottom part.

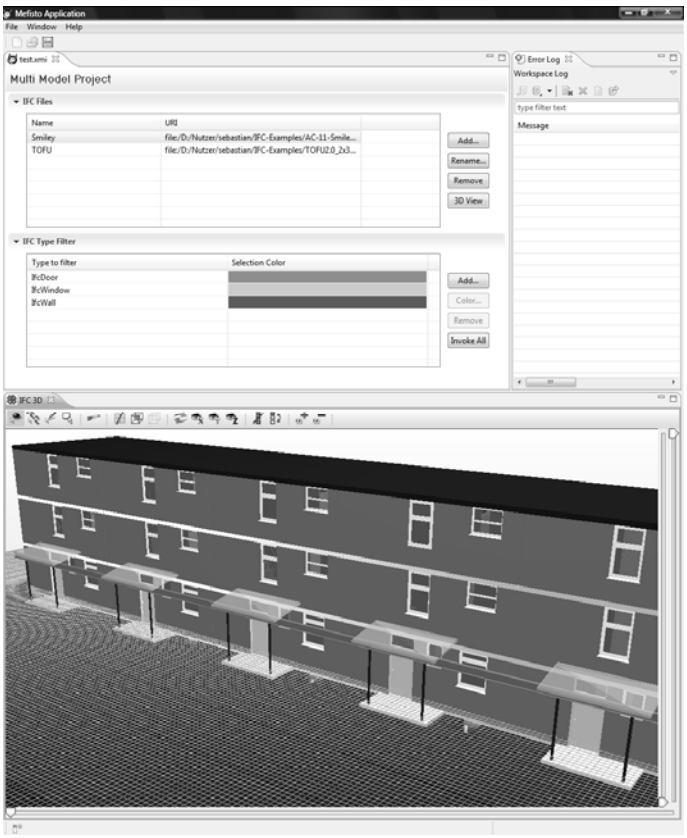

Figure 3. GUI of the framework with an example set of IFC related plug-ins

In general, client modules are responsible for the resource management themselves. This means that they must provide a mechanism to load and manage instances of a certain single model on their own (objective 4). The framework already comes with plug-ins for managing IFC documents in SPF format (ISO 10303-21), GAEB-XML (GAEB 2009) and Microsoft Project XML (cf. http://msdn.microsoft.com/en-us/library/). Support for further model types is planned. IFC-model support is provided by Open IFC Tools (OIT 2009). This is also used for the implemented IFC 3D viewer (cf. Theiler et. al. 2009).

The framework defines an extensible data model for multi-model projects. Client plug-ins can provide their persistent data by implementing a *marker interface*. In this way they can take part in the frame-

work's global load-edit-save cycle. The data can consist of single model instances, references to them and any data which is necessary to restore the state of a plug-in. Given that functionality, the data model of the framework can be seen as a universal container. Hence, it is a *multi-model container*, too (objective 5).

The framework expects persistent data to be created by the use of the Eclipse Modeling Framework (EMF). EMF is a toolset to produce source code out of structured models (Steinberg et al. 2009). It provides different serialisation mechanisms from which XMI was chosen for the framework's container persistence. Thus clients don't have to implement persistence themselves – instead they can rely on a consistent framework service.

Figure 4 shows a XML fragment of an example multi-model project's serialisation. The element *mf_ifc:IfcContainer* represents the persistent data of the plug-in which is responsible for holding IFC documents (e.g. as SPF or IfcXML files). The element contains a collection of such IFC documents that are relevant to the current project. They are stored by an URI-reference to their actual files. The element *mf_type_filter:TypeFilterContainer* holds the data of an IFC-filter plug-in. In this case this is a collection of several filters for one IFC-type each and their dedicated selection color for the IFC 3D viewer.

```
<?xml version="1.0" encoding="ASCII"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/x
  <mf_ifc:IfcContainer>
    <documents name="Smiley" uri="file:/D:/Nutzer/sebastia
    <documents name="TOFU" uri="file:/D:/Nutzer/sebastian/
  </mf_ifc:IfcContainer>
  <mf_type_filter:TypeFilterContainer>
    <typeFilters ifcTypeToFilter="IfcDoor">
      <selectionColor red="128" green="128" blue="255"/>
    </typeFilters>
    <typeFilters ifcTypeToFilter="IfcWindow">
      <selectionColor red="128" green="255" blue="128"/>
    </typeFilters>
```

Figure 4. XMI serialisation fragment of an example multi-model project

4 PRINCIPAL FRAMEWORK USE

A client plug-in must be registered to the framework by a specifically provided *extension point*. It has to point to a class which implements the interface *IMfPlugin*. This type is the entry point for the data structure of the client plug-in. At runtime there will be one instance for each opened multi-model project. The interface signature is shown in Figure 5.

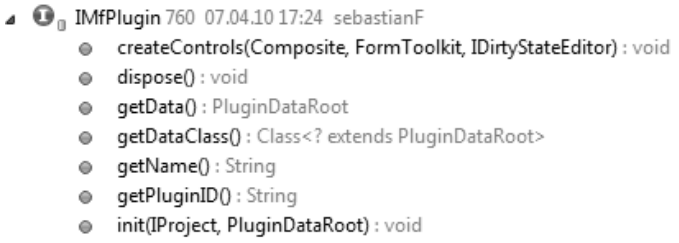

Figure 5. Signature of the *IMfPlugin* interface

To explain the basic concept of the framework's use, the semantics of the four most important methods of the interface are shortly described below.

getPluginID()
This method will return a unique string (in the range of all client plug-ins), which is used to identify that client plug-in. Thereby, instances of further specific plug-ins can be found and accessed in the scope of the current multi-model project.

getDataClass()
If the plug-in offers persistent data, the method will return the concrete class which contains the entire data. This class must implement the marker interface *PluginDataRoot*. It is used by the framework to recognize the dedicated plug-in of a container's data element. The actual data can be accessed by calling *getData()*.

init(…)
This method is called by the framework at the start of the plug-in's lifecycle. It is passed its context in form of the current multi-model project (*IProject*) and, if exists, the loaded persistent data of the last session. At the end of the lifecycle, e.g. at program exit, *dispose()* is called by the framework, where clients can release beforehand allocated resources.

createControls(…)
Due to the Eclipse lazy loading mechanism, a plug-in's user interface must not necessarily be rendered at the start of its lifecycle. Therefore this method is called when user interface controls are shown up first time. Clients can set up their user interface here, which is shown in the editor part inside a so called section. The name of the section is obtained by the method *getName()* and should represent the plug-in's name.

5 CLIENT EXTENSIONS

An important intention of the developed multi-model framework is to encourage clients to reuse and extend existing plug-ins. Thus, clients can make use of additional software libraries to achieve more efficient results. This means that a plug-in can be an information consumer and provider at the same time. Clients should respect that and provide their main functionality to others.

Plug-in implementers are free in their use of provided data, e.g. the end user selection of model instances in a table. In that way, a change management plug-in for IFC could interpret a single selected IFC document as basis for comparison against all other loaded ones, while it does not consume other provided models. On the other hand, a plug-in which

links construction site data with time schedules automatically could just be able to make use of both loaded model instances. It would leave further instances unused, regardless of their selection state. However, client implementations should explain their behaviour and prerequisites to other clients and the end users in unambiguous way.

IFC-based plug-in for multi-model filtering

Figure 6 on the right presents an example of the IFC-based plug-in for multi-model filtering developed in the Mefisto project by extending and adapting the original tool described in (Theiler et al. 2009). It enables definition of filters on class and instance level, combined application and visualisation of multiple filters on multiple IFC and non IFC model instances, comparison of filters by means of multi-colour visualisation and so on. Moreover, via its integration in the proposed framework it can be used for a variety of purposes by other construction management services, such as scheduling, cost calculation and controlling.

Further multi-model plug-ins

Due to the genuine openness of the platform, various other single or multi-model plug-ins can be envisaged. In fact, using the described approach a number of such plug-ins is currently being developed or designed in the Mefisto project. Table 1 provides an overview of some of these client plug-ins and their related multi-model space.

Depending on the specific data and process context the actual information containers comprising filtered model instances of one, some or all of the shown models are assembled and passed to the requesting client plug-in and/or produced as output in uniform manner. Hence, while the underlying domain models can be quite different in structure, using the container approach and the basic framework functionality consistent multi-model interoperability can be achieved, provided that the required matching and mapping problems are adequately handled.

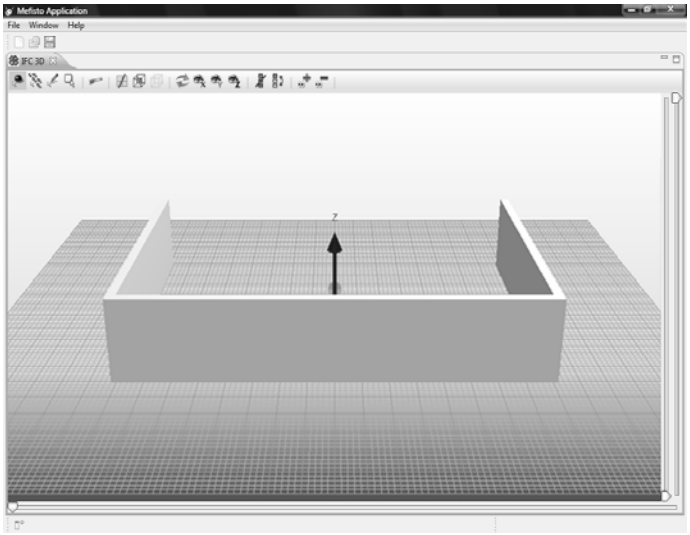

Figure 6. Screenshot of the IFC-based plug-in showing the result of a filter applied to the IFC model (walls, spaces) and a non IFC scheduling model (planned date of wall assembly).

Table 1. Client Plug-ins in the Mefisto project and the usage of their related single-models as input (In), output (Out) or both (I/O). Model processing shown in parenthesis is optional.
GRANID, RIB iTWO and Plant Simulation are commercially available tools that are being integrated in the Mefisto platform. GRANID is a management and controlling system developed by the project partner GIB Greiner GmbH, RIB iTWO is a new specification, cost calculation and scheduling system developed by the project partner RIB AG, and Plant Simulation is a widely known simulation tool of the Siemens AG.

Client Plug-Ins \ Models	Process	BIM	Construction Site	Organisation	Specifications	Costs	Schedules	Risks & Uncertainties
Dynamic management of construction processes	I/O	(In)	(In)	(In)			Out	
Construction site configurator		In	Out	(In)		(In)		
Controlling and risk management	In	In		In	I/O	I/O	I/O	Out
Configuration and adaptation of simulation models	In	In	In	In		(In)	(In)	
Strategy and scenario manager for simulation tasks	In	In	I/O	In	I/O	(In)	(In)	(In)
Collision checker for supply and assembly tasks	In	In	In	(In)			(In)	
GRANID Plug-In	In	In			I/O	I/O	I/O	Out
RIB iTWO Plug-In	I/O	In		(In)	I/O	I/O	I/O	
Plant Simulation Plug-In	In	In	In	(In)		(In)	(In)	(In)

6 SUMMARY AND FUTURE PROSPECTS

The developed multi-model framework is seen as a relevant approach to foster ICT support for construction processes. It was developed as an extensible basis for upcoming multi-model investigations. Due to its predefined generic character, all functionality must be provided by client plug-ins. However, for convenience, various model management and visualisation plug-ins are built in. Necessarily, research must progress at least in the fields of single-model association and multi-model filters to better cope with the multi-model challenges in AEC/FM.

The framework has an implementation of a general multi-model container. This is seen as an elementary step and a precondition for further research in that direction.

The use of established frameworks like Eclipse and EMF and the developed design of a small and flexible extension interface enable clients to write plug-ins by less effort. Hence, developers can concentrate on conceptual work instead of developing test environments themselves.

Though the framework is already ready for use, further development is planned. Envisaged new features are metadata generation, web service support and ontology integration.

ACKNOWLEDGEMENTS

The research work presented in this paper is part of the Mefisto project which is funded by the German Ministry of Education and Research (BMBF) under Grant No. 01IA09001 and by the project partners. This support is gratefully acknowledged.

REFERENCES

Aouad, G., Lee, A. & Wu, S. (eds.) 2007. *Constructing the future: nD modelling*. Taylor & Francis, London, UK.

D'Anjou, J., Fairbrother, S., Kehn, D., Kellerman, J. & McCarthy, P. 2006. *The Java Developer's Guide to Eclipse*. 2nd Ed., Addison-Wesley.

Froese, T. 2003. *Future directions for IFC-based Interoperability*. ITCon, Vol. 8, Special Issue "IFC - Product models for the AEC arena", pg. 231-246, Available at: http://www.itcon.org/2003/17.

GAEB 2009. *Gemeinsamer Datenausschuß Elektronik im Bauwesen – GAEB-DA-XML Version 3.1*. Available at: http://www.gaeb-da-xml.de/

Gamma, E. & Beck, K. 2003. *Contributing to Eclipse*. Addison-Wesley Professional.

ISO 10303-21. 2002. *Industrial automation systems and integration -- Product data representation and exchange -- Part 21: Implementation methods: Clear text encoding of the exchange structure*. © International Standards Organisation, Geneva.

Keilholz, W., Ferries, B., Andrieux, F. & Noel, J. 2009. *A simple, neutral building data model*. In Zarli, A. & Scherer, R. J. (eds.) "eWork and eBusiness in Architecture, Engineering and Construction", Proc. 7th European conference on product and process modelling, Sophia Antipolis, 10-12 September 2008, CRC Press / Balkema, pp. 105-109.

Kiviniemi, A., Tarandi, V., Karlshoj, J. Bell, H. & Karud, O. J. 2008. *Review of the Development and Implementation of IFC Compatible BIM*. © Erabuild 2008, 128 p.

OIT 2009. *Open IFC Tools*. Available at: http://www.openifctools.org

Scherer, R. J., Schapke, S.-E. & Katranuschkov, P. 2010. *Mefisto: A Model, Information and Knowledge Platform for the Construction Industry* (in German: "Mefisto: Eine Modell-, Informations- und Wissensplattform für das Bauwesen"), Project Presentation, BMBF Project 01IA09001.

Schapke, S.-E., Katranuschkov, P. & Scherer, R. J. 2010. *Ontology-based ICT platform for management, simulation and decision making in large scale construction projects*. To appear in: "Proc. ICCCBE 2010 Int. Conference", Nottingham, UK, 6 p.

Steinberg, D., Budinsky, F., Paternostro, M. & Merks, E. 2009. *EMF: Eclipse Modeling Framework*, 2nd ed., Addison-Wesley.

Theiler, M., Tauscher, E., Tulke, J. & Riedel, T. 2009. *Visualisation of IFC objects using Java3D* (in German: "Visualisierung von IFC-Objekten mittels Java3D"). In: von Both, P. & Koch, V. (eds). "Forum Bauinformatik 2009". Karlsruhe: Universitätsverlag.