Development of Information Delivery Controlling Tool based on Process Modeling

Zhiwei Meng, meng@uni-wuppertal.de
Chair of Construction Management & Economics, University of Wuppertal, Germany

Noemi Kremer, kremer@dc.rwth-aachen.de
Chair of Design Computation, RWTH Aachen University, Germany

Brian Klusmann, klusmann@uni-wuppertal.de
Chair of Construction Management & Economics, University of Wuppertal, Germany

Anica Meins-Becker, a.meins-becker@uni-wuppertal.de
Chair of Construction Management & Economics, University of Wuppertal, Germany

Jakob Beetz, j.beetz@caad.arch.rwth-aachen.de
Chair of Design Computation, RWTH Aachen University, Germany

Abstract

With rapid development of Building Information Modeling (BIM) technology, various contributions were made to the realization of a precise examination of required information at each step of workflow throughout the lifecycle of projects. However, a BIM based tool for inspection on general information in the working process, such as **when** should **who** delivery **what** information, has rarely been taken into consideration. This paper firstly takes a brief look at related instruments for process management and information controlling. After the framework of a web-based tool for information delivery controlling is introduced, a tool configured with a rulesets-generator and embedded with an existing rulesets-checker is implemented. Two BIM use cases were chosen as instances for validating and adjusting the information-controlling tool. Finally, the results of the prototypical implementation are published as open-source and the restrictions as well as future developments of the information delivery controlling tool are discussed.

Keywords: Process Modeling Database, Exchange Requirements, BIM Use Cases, Model View Definition, Information Delivery Manual

1 Introduction

While the applying of BIM in Architecture, Engineering, and Construction (AEC) industry is increasingly widespread, higher requirements for the development of BIM applications are met, which include the inspection of delivered documents at each section of project phases. For instance, the concept and technical implementation of model view definition (MVD) enables the mappings of information in exchange requirements at each defined point of data drop with IFC elements to meet particular exchange scenarios (Scheffer et al 2018). These technical implementations build up a steady foundation for information controlling during project development. However, the enforcement of BIM in which numerous parties are involved with heterogeneous views and interests requires a clear definition of roles and responsibilities (Scheffer et al 2018). Therefore, the development of a tool that takes both the working process of all project participants and the exchange requirements at each stage into account is urgently needed.

2 Existing theories & previous work

2.1 The database of process modeling and use cases

With the constant accumulation of BIM use cases in real projects, the Use Case Management Service (UCMS) suggests that the need for establishing common, machine-readable and reusable repositories of use cases with their corresponding exchange requirements is rather large (buildingSMART International 2021). Since 2016, the process-modeling database at University of Wuppertal (BUW database) has been continuously enriched with working processes from the perspective of execution as well as of technical approaches under the circumstances of using BIM. At present, it contains about 30 BIM use cases gathered from research and practical projects in the AEC industry. For each BIM use case, the workflow is described with the format of Business Process Management Notation (BPMN), and is ready to answer the following questions: When (time) does who (process owner) process what (information input), after which (other applicable) information, why (process goal), to what (information output)? (Helmus et al 2017) The database is constructed with five levels as shown in a reduced form in Table 1. For instance, the required working process and exchange documents or files are contained at level three, and their concrete exchange requirements demanding by the generation of a document are stored at level five.

Table 1. Structure of the University of Wuppertal database

BUW process database					
Process L1	Project phase, Target				
	Process L2	Process owner, Target			
		Process L3	Process owner, Input, Output, Other applicable information, Associated use case, Possible object types		
			Process L4	Input, Output, Other applicable information, Property Sets	
				Process L5	Input, Output, Other applicable information, Properties

Since the BUW database is built upon concrete examples and contains extensively defined information requirements, it can serve as a basis for the proof of information controlling.

2.2 Linkage of BUW process database with BuildingSMART Data Dictionary

For international collaboration, the buildingSMART data dictionary (bSDD) was implemented based on ISO12006. It collects terms and descriptions in multiple languages and domains. To avoid duplicate creation of entities or attributes in BUW database as much as possible, a general valid designation and description of terms is required. Therefore, the buildingSMART Data Dictionary (bSDD) version 4 was used. The basic entities and attributes from the IFC classification were first imported into the BUW database via API from bSDD and compared with the BUW elements. The BUW elements that have the same name as in the bSDD will be overwritten.

The new version 5 of the bSDD API exists in private beta and will correspond to the updated version of the underlying data model standardized in the ISO12006-3 (buildingSMART International 2020). In the future, newly created BUW characteristics can be sent as a request to the bSDD during an alignment to add them to the bSDD.

2.3 Information Delivery Manual

In order to describe the concrete information about who delivers what to whom at which point during the life cycle of a project, the information delivery manual (IDM) standard was introduced by buildingSMART (Beetz et al 2018). The IDM thus plays an important role in specifying information requirements in tenders, for contractors and for guidance in projects (CEN/TC 442 2019). According to ISO 29481-1, the primary IDM components include use case (UC), process map (PM), exchange requirement (ER) and model view definition (MVD) (see Figure 1). The user, such as the client, can generate an IDM by defining strategic goals, determine necessary information requirements and get machine-readable file as mvdXML.

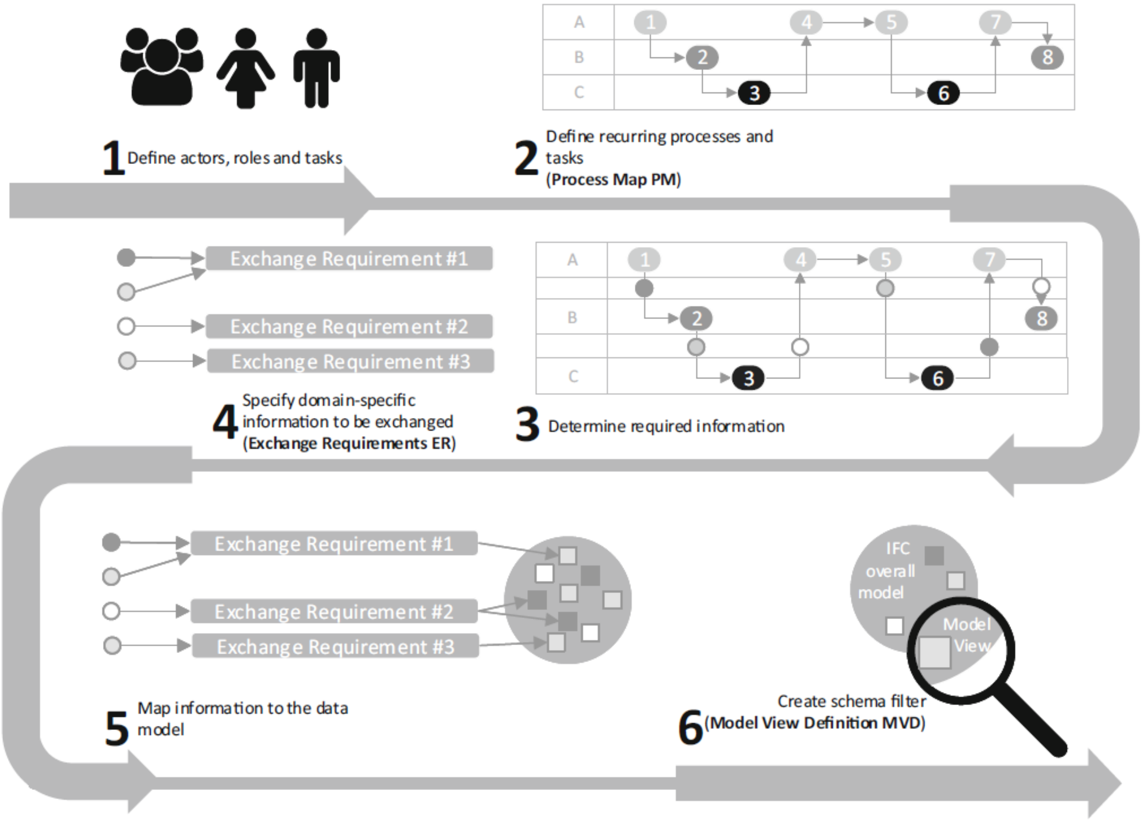

A
B
C
1
2
3
4
5
6
7
8
1 Define actors, roles and tasks
2 Define recurring processes and tasks (Process Map PM)
Exchange Requirement #1
Exchange Requirement #2
Exchange Requirement #3
4 Specify domain-specific information to be exchanged (Exchange Requirements ER)
3 Determine required information
IFC overall model
Model View
5 Map information to the data model
6 Create schema filter (Model View Definition MVD)

Figure 1. Process of generating IDM and its assignment to various requirements (Beetz et al 2018)

However, due to the complexity of developing IDM content and specification, it is still difficult to implement or share IDMs under the current specifications (Jeon & Lee 2018). To solve these issues, the idmXML schema is still under development and is supposed to be published in ISO 29481-3.

2.4 MVD checking tools

The Open MVD Model checker tool is based on the open-source software framework bimserver.org (Oraskari 2020). It is a generic tool for verifying IFC model content for defined exchange requirements using mvdXML (RWTH-Aachen 2020). After successful verification, a report is returned as a BCF file, containing all detected issues. Furthermore, no additional plugins for mvdXML validation and BCF files are needed. An integrated 3D Viewer for visualized results is currently not available. In addition, the MVD Model Checker exposes a public REST OpenAPI (RWTH-Aachen 2020). Such standards based IFC model checking tools are also available in third-party applications such as a SimpleBIM application by Datacubist (Simplebim 2020) or the open source XBIM Xplorer by XBIM toolkit (Xbim 2020).

In this project, the open MVD Model checker is the preferable tool due to its design as a portable open-source web service containing an integrated OpenAPI, the open-source availability and provides of BIM integrable report files (see Table 2).

Table 2. Comparison of MVD Model Checker, XBIM Xplorer and SimpleBIM

	MVD Model Checker	XBIM Xplorer	SimpleBIM
Tool by	generic tool by RWTH -OpenAPI	XBIM toolkit	Datacubist
Availability	open-source web service and available for download	.NET Windows open-source application - available for download	not open-source available for download
IFC	IFC2x3 and IFC4	IFC2x3 and IFC4	IFC2x3 and IFC4
mvdXML	mvdXML V1-1, V1.1 and V1.2draft 3	Plugin for mvdXML	mvdXML
Output file	JSON and BCF file report	3D Viewer (Plugin for BCF file)	3D Viewer and BCF file

3 Methods

To develop a tool for information delivery controlling based on the process modeling, the current problems in relation to information controlling were discovered. Therefore, the specific characters of an optimized tool were collected through a workshop. Afterwards, a relative tool with prioritized functions from the workshop has been developed. Finally, the tool was tested using two specific use cases and optimized with feedback from the testing results.

In the workshop, we invited 11 practitioners representing stakeholders from AEC industry, such as project development (1), project management (3), architecture (2), civil engineer (1), building operation (2) and building management (2), to nominate and prioritize the possible functions of an information controlling system by considering their real working scenarios. Requirements for the tool were classified according to the MoSCoW method (Clegg & Barker 1994). The results of “Must have” and “Should have” are shown in the following table (see Table 3).

Table 3. Results of MoSCoW prioritization of product requirements for the demonstrator

Product requirements for the Information Delivery Controlling Tool	
Must have	**Should have**
A database of templates for Employer’s Information Requirements (EIR) based on existing use cases is developed for the swift generation of project requirements.	For end users, it should be possible to edit, evaluate and visualize the data sets through a portal.
The EIR database stores attributes (including geometric), rulesets, etc. related to corresponding use cases.	Filter and search functions are provided for a better overview and usability.
Checking tools are based on open standards and interface.	A web-based portal is provided to search for EIR templates according to required criteria and characteristics (e.g. use case).
The data sets should be able to be reflected differently depending on the use cases, but in a way that the users can understand.	Object template database should be taken as reference for exporting the model as an open exchange format IFC.
It should be a database-integrated tool that the data requirement can be read in its model checker for automatic comparison.	Develop a web-based object configurator.

	A database-integrated tool allows the automatic examination of EIR which is agreed by participants against the generated version from templates.

According to the listed product requirements, a web-based comprehensive tool is in high demand, which provides integration of the ER database as well as editable data sets, and contains the function of automatic checking based on open standards.

Combing requirements in practice, a process flow for an information delivery controlling system is proposed: Initially, the project owners get a text-based report as well as a table of attributes by choosing one or more BIM use cases from the UC/ER database. Once the table of ER is completed, it is transferred to the contractors so that they can start planning and bringing those prerequisites into their BIM models. The BIM models can then be exported as IFC file through authoring software and sent to BIM manager for model checking. After receiving the MVD file from the customer and the IFC model from the contractor, a comparison between the requested and the delivered information can be enforced by the BIM manager. Finally, the checking results are presented to the contractor for a revision of the model, or to the customer for the final report of success.

For a seamless process of creating and checking against rulesets, various machine-readable file formats should be embedded. The proposed Information Delivery Controlling (ILC) system provides interfaces and compatibility with a number of machine-readable, standardized file formats including MVD, IFC, BCF, CSV (Figure 2).

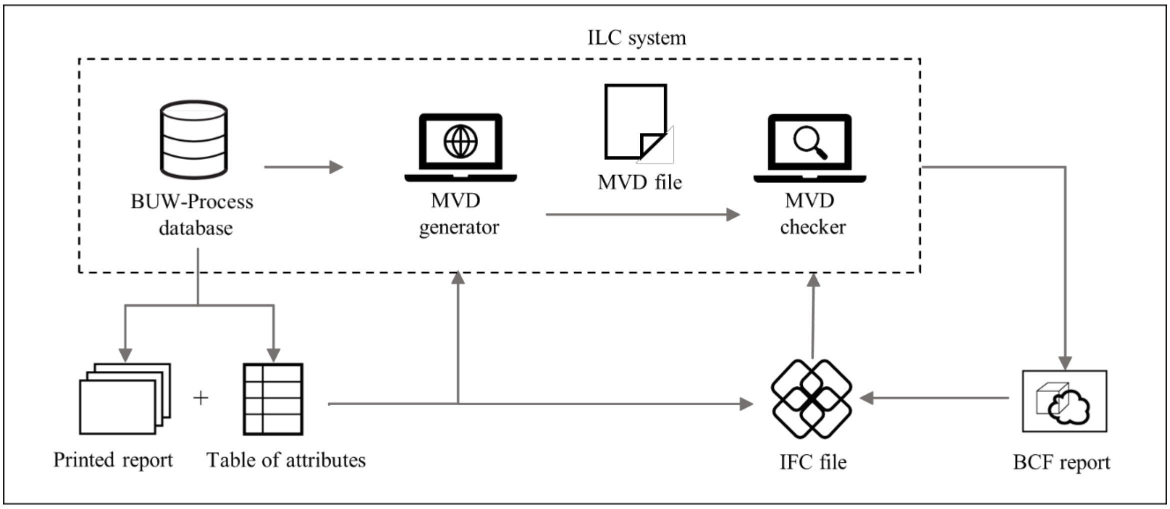


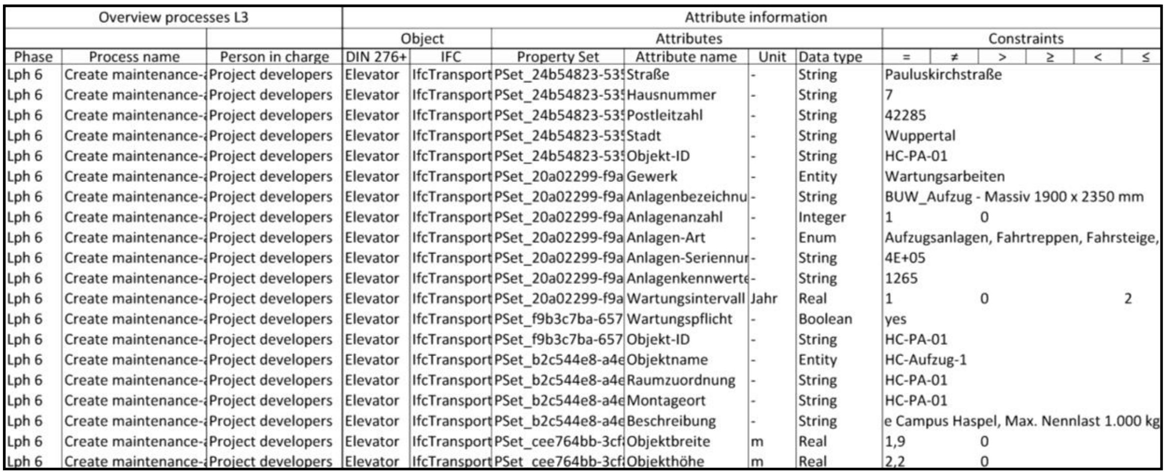
Figure 2. Mechanism of ILC system

4 Findings

Based on the mechanism of ILC system, an interoperable service between UC/ER database and Checker tool has to be achieved for a reliable workflow. During the development of the ILC tool, two use cases, maintenance management and cleaning management, were chosen for simulating the real usage environment. In this paper, the use case of cleaning management will be taken as an example to reveal the implementation results.

4.1 Getting exchange requirements from the database

The table generated from the ER/UC database marks the first transfer point, as it is the basis for the MVD generator to generate mvdXML. In the database, exchange requirement tables are defined and stored. The process name, person in charge, relevant object, related attributes and property sets are contained in the exchange requirement table. To check the plausibility of the value for the attribute, general conditions, such as unit, data type, constraint, cardinality are also specified (see Figure 3).



Overview processes L3			Attribute information											
			Object		Attributes				Constraints					
Phase	Process name	Person in charge	DIN 276+	IFC	Property Set	Attribute name	Unit	Data type	=	≠	>	≥	<	≤
Lph 6	Create maintenance-	Project developers	Elevator	IfcTransport	PSet_24b54823-535	Straße	-	String	Pauluskirchstraße					
Lph 6	Create maintenance-	Project developers	Elevator	IfcTransport	PSet_24b54823-535	Hausnummer	-	String	7					
Lph 6	Create maintenance-	Project developers	Elevator	IfcTransport	PSet_24b54823-535	Postleitzahl	-	String	42285					
Lph 6	Create maintenance-	Project developers	Elevator	IfcTransport	PSet_24b54823-535	Stadt	-	String	Wuppertal					
Lph 6	Create maintenance-	Project developers	Elevator	IfcTransport	PSet_24b54823-535	Objekt-ID	-	String	HC-PA-01					
Lph 6	Create maintenance-	Project developers	Elevator	IfcTransport	PSet_20a02299-f9a	Gewerk	-	Entity	Wartungsarbeiten					
Lph 6	Create maintenance-	Project developers	Elevator	IfcTransport	PSet_20a02299-f9a	Anlagenbezeichnu	-	String	BUW_Aufzug - Massiv 1900 x 2350 mm					
Lph 6	Create maintenance-	Project developers	Elevator	IfcTransport	PSet_20a02299-f9a	Anlagenanzahl	-	Integer	1		0			
Lph 6	Create maintenance-	Project developers	Elevator	IfcTransport	PSet_20a02299-f9a	Anlagen-Art	-	Enum	Aufzugsanlagen, Fahrtreppen, Fahrsteige,					
Lph 6	Create maintenance-	Project developers	Elevator	IfcTransport	PSet_20a02299-f9a	Anlagen-Seriennur	-	String	4E+05					
Lph 6	Create maintenance-	Project developers	Elevator	IfcTransport	PSet_20a02299-f9a	Anlagenkennwerte	-	String	1265					
Lph 6	Create maintenance-	Project developers	Elevator	IfcTransport	PSet_20a02299-f9a	Wartungsintervall	Jahr	Real	1		0			2
Lph 6	Create maintenance-	Project developers	Elevator	IfcTransport	PSet_f9b3c7ba-657	Wartungspflicht	-	Boolean	yes					
Lph 6	Create maintenance-	Project developers	Elevator	IfcTransport	PSet_f9b3c7ba-657	Objekt-ID	-	String	HC-PA-01					
Lph 6	Create maintenance-	Project developers	Elevator	IfcTransport	PSet_b2c544e8-a4e	Objektname	-	Entity	HC-Aufzug-1					
Lph 6	Create maintenance-	Project developers	Elevator	IfcTransport	PSet_b2c544e8-a4e	Raumzuordnung	-	String	HC-PA-01					
Lph 6	Create maintenance-	Project developers	Elevator	IfcTransport	PSet_b2c544e8-a4e	Montageort	-	String	HC-PA-01					
Lph 6	Create maintenance-	Project developers	Elevator	IfcTransport	PSet_b2c544e8-a4e	Beschreibung	-	String	e Campus Haspel, Max. Nennlast 1.000 kg					
Lph 6	Create maintenance-	Project developers	Elevator	IfcTransport	PSet_cee764bb-3cf	Objektbreite	m	Real	1,9		0			
Lph 6	Create maintenance-	Project developers	Elevator	IfcTransport	PSet_cee764bb-3cf	Objekthöhe	m	Real	2,2		0			

Figure 3. A part of exchange requirement table from BUW database (screenshot)

The datasets can be exported as CSV file, and is editable by the end users. Since these files provide merely a baseline of requirements. More project-specific defined object types, attributes or constraints shall be supplemented to these editable files by the project owners.

4.2 MVD generator

The generation of a computer readable MVD is based on utilizing predefined mvdXML 1.1 template files. An important aspect of this procedure is the reduction of the effort in generating mvdXML templates by limiting their scope of application on property set definitions. Therefore, the MVD generator is exactly adapted to the ER table of the ER database. The process of generating mvdXML itself is carried out in 4 steps (Figure 4):

1. Reading the provided data defined as excel or csv file and extracting all necessary information, such as outlined in Figure 4.
2. Translating the extracted information into structured inputs for the mvdXML 1.1 template files (see Figure 5).
3. Filling the predefined placeholders in the provided template file with translated data.
4. Generating the output files as mvdXML 1.1. In order to keep a process-oriented generation of mvdXML 1.1 files, a new file is generated at each change of process owner.

As a result, the generation of mvdXML is process owner bounded, so that information to be checked can be assigned to a person owner. The generated mvdXML(s) can be further used for validating required IFC model content.

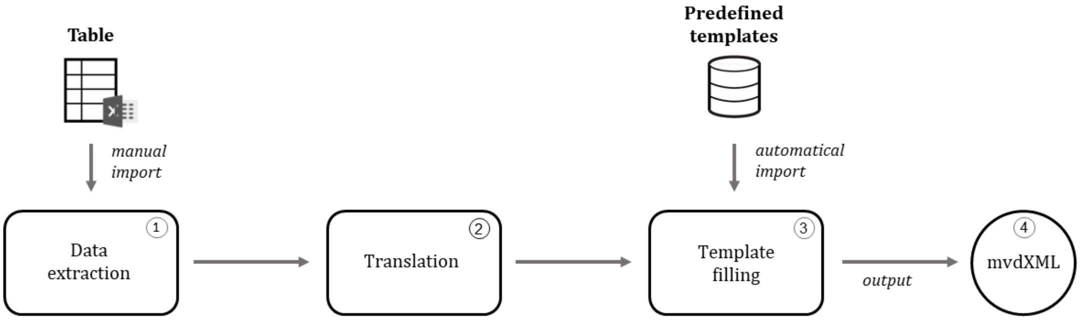


Figure 4. Process steps of generating mvdXML

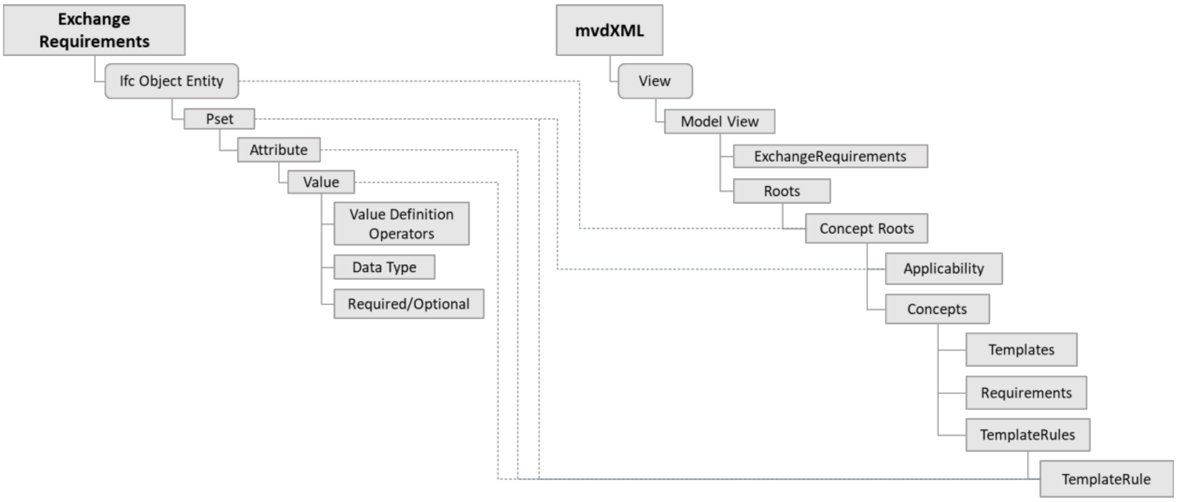


Figure 5. Mapping between data in ER and mvdXML

In the use case cleaning management three participants (owner, contractor, facility manager) take responsibilities in the working process. When there is a change in process responsibility, a new information request is generated, with the technical part recorded in a mvdXML. Afterwards the IFC objects assigned to the participant are identified; in this use case, relevant IFC entities are IfcWindow and IfcSlab. For these in turn the assigned property sets are filtered out. Furthermore, for each identified property set, all associated properties are filtered iteratively. This process can be continued up to the value definition.

4.3 Validation of IFC file with MVD Checker

In advance of the checking process an IFC model has to be prepared. The MVD model checker API has two inputs, the IFC file and mvdXML file and two outputs, the BCF file and the JSON file. In general, the process of checking IFC files using mvdXML is carried out in four steps (Figure 6):

1. Passing the required data files one IFC and one mvdXML to the MVD model checker API.
2. Preliminary examination of all IFC object elements for required property sets assigned.
3. Main examination of identified IFC object elements for required properties and value definitions. A comparison between the requested and the delivered information with the following points to check for:
    a. Is the requested information available?
    Here, a comparison is made to see whether the information requested by the client is available in the designated place within the file. Observance of the modeling specifications is necessary for this, so that a (semi-)automated comparison can take place.
    b. Is the stored information plausible?
    If the requested information is available according to the 1st check, it can be checked for plausibility.
4. After successful validation the test results are returned as JSON and BCF ZIP formats. The JSON file can be displayed on the user interface considered a traffic light feedback function. The BCF ZIP file can be downloaded for further IFC model improvements.

The MVD Checker just checks for the delivered information in an IFC model, it does not check who the process owner is or when the requested information should be delivered.

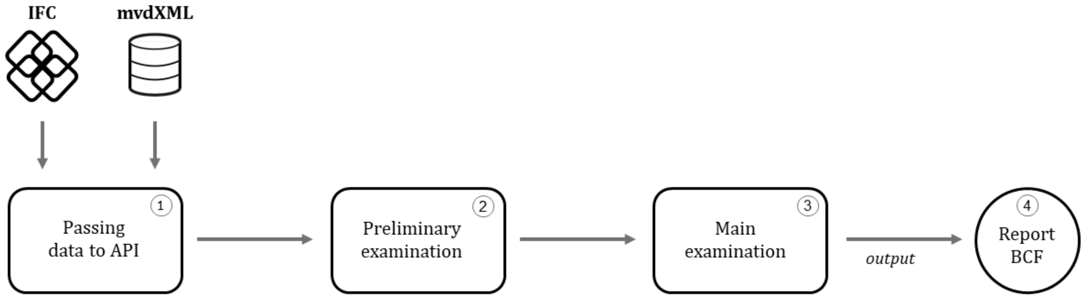


Figure 6. Process steps of validating IFC file with MVD Checker

In case of the use case "cleaning management" the IFC file and mvdXML file are passed to the checker. During preliminary examination all IfcWindow and IfcCovering elements are checked for required property sets assigned, to pre-filter the elements in order to perform the actual checking. Because not all of the objects sub-classed from IfcCovering are relevant to cleaning management. For instance, the ceiling is irrelevant to cleaning process, and therefore should be filtered out. Those filtered IFC object elements are further checked for required attributes and value definitions, such as the Object-ID or the Room-ID. Finally, the checking result are available as JSON and BCF file report.

4.4 Discussion

A process-oriented information delivery controlling system based on open source technology was developed and the essential modules were implemented. The tabular connection between process owner and exchange requirements in the database is taken up when creating the mvdXML. The system allows to assign the respective actors who have to provide which information on a process-related basis. Supplied IFC models can be checked for the required contents using mvdXML. Negative checking results are recorded using the open source format BCF. This supports improvement work on the IFC model through feedback into appropriate software, until an error-free test result is achieved. In the context of user-friendly operation, a user interface has been implemented were both, the mvdXML generator and the MVD model checker, can be operated.

Although the datasets from UC/ER database are successfully mapped with data structure of mvdXML, the database is only loosely coupled to the MVD generator and checker. Therefore, a workflow without media discontinuity in this system is not yet possible and is addressed as part of future work towards standards based microservices architectures in the context of evolving CDE frameworks. In addition, there is currently no control to fill an exchange request table with individual, corresponding values for each attribute, ensuring that only the intended data type can be entered. Furthermore, with using the ILC tool only the content and author of delivered model can be controlled. The delivery time and other important process-related criteria, will be integrated into future versions of the framework.

5 Conclusion

5.1 Limitations

Two use cases were used in the implementation process for validation. In order to control and iteratively validate the progress of a demonstrative process, user-relevant data sets are important and the ILC system should be further tested with a wider variety of use cases. It can be assumed, that further use cases will reveal other possibilities for improvement. As the implementation process in section 4 shows, although role assignments to stakeholders for delivered information have been considered and are examined by the ILC system, the checking on delivery time is still not included. This means the checking of whole workflow is only partially accomplished.

5.2 Concluding remarks

In this paper, an information delivery controlling tool considering the participation of all shareholders has been developed and validated with two use cases. A number of practical issues for an interoperable service-oriented architecture between UC/ER database and MVD Model Checker have been considered as well. For example, when the automatic creation of EIR was implemented, the way of importing the requirements into authoring software has been thoroughly tested as well. The results of implementation are published on https://github.com/Design-Computation-RWTH/ILC_Demonstrator, and can be further enriched or developed.

5.3 Future works

To achieve a fully automatic information delivery controlling, there are still some difficulties to address and solve: bringing a process management tool into ILC system, such as the module of deadline checking from CDE, or integrating the module of checking on data format idmXML into ILC platform, as the idmXML file should contain the information about delivery date as well.

Acknowledgements

We gratefully acknowledge the support of Federal Office for Building and Regional Planning (BBSR) of Germany for funding the project. We sincerely thank all our practice partner for their participants in workshops and providing advice according to their experience.

References

Beetz, J., André Borrmann, & Matthias Weise. (2018). Process-based definition of model content. In Building Information Modeling (pp. 127–138). Springer.

buildingSMART International. (2020). buildingSMART Data Dictionary. Retrieved from https://www.buildingsmart.org/users/services/buildingsmart-data-dictionary/

buildingSMART International. (2021). Use Case Management. Retrieved from https://ucm.buildingsmart.org/

CEN/TC 442. (2019). First WD WI 442023 CEN/TR Guidance for understanding and using EN ISO 29481-1.

Clegg, D., & Barker, R. (1994). Case method fast-track: a RAD approach. Addison-Wesley Longman Publishing Co., Inc.

Helmus, M., Meins-Becker, A., Kelm, A., Bodtländer, C., Kaufhold, M., Kesting, H., … Zibell, M. (2017). TEIL 1: Grundlagenbericht Building Information Modeling und Prozesse.

Jeon, K., & Lee, G. (2018). Information Delivery Manual (IDM) Configurator: Previous Efforts and Future Work. In 18th International Conference on Construction Applications of Virtual Reality.

Oraskari, J. (2020). OnlineMvdXMLChecker.

Pauweis, P., McGlinn, K., Törmä, S., & Beetz, J. (2018). Linked data. In Building Information Modeling (pp. 181–197). Springer.

RWTH-Aachen. (2020). mvdXMLChecker OpenAPI Interface. Retrieved from http://lbd.arch.rwth-aachen.de/mvdXML-Checker/apidocs/.

Scheffer, M., Mattern, H., & König, M. (2018). BIM project management. In Building Information Modeling (pp. 235–249). Springer.

Simplebim. (2020). Simplebim-Features. Retrieved from https://simplebim.com/features/

Xbim. (2020). Xbim Xplorer. Retrieved from https://docs.xbim.net/downloads/xbimxplorer.html