ifcModelCheck

A tool for configurable rule-based model checking

Sebastian Ebertshäuser [1], Petra von Both[2]
Karlsruhe Institute of Technology (KIT) Germany
http://blm.ieb.kit.edu
[1]sebastian.ebertshaeuser@kit.edu, [2]petra.vonboth@kit.edu

Abstract. *On behalf of the BBR (German Federal Office for Building and Regional Planning) the development of an Industry Foundation Classes (IFC) based inspection tool was accomplished as application on an underlying work-in-progress development framework. By providing a machine-based checking process the tool ModelCheck was rolled out to meet demands emerged during pilot projecting. Thus it is capable of processing automated compliance checks on quality criteria for the authorities, e.g. documentation guidelines of BBR regarding building and real estate documentation or building information modeling (BIM) quality criteria formed for the Humboldt-Forum project – a BIM pilot-project managed by BBR. ModelCheck supports checks on IFC models - formal against schemes and logical inspection with regards to alpha-numeric content by using xml-based configurable rules.*
Keywords. *BIM; quality assurance; rule-based model checking; collaboration*

INTRODUCTION

Besides being a promising concept from a general point of view, building information modeling (BIM) in real world is still confronted with problems in terms of overarching business and process related co-operation on a base of its models. The results of a market analysis regarding potentials and hindrances of BIM application in Germany identify great prejudice and reticence coming to business overarching transmission and cooperative usage of BIM models (von Both, 2012). In the market the benefit of BIM in terms of co-operation with project partners is worse than other more operative aspects by far. In such co-operation activities nowadays one reason, besides inadequate technical interfaces, seems to lay in insufficient specification of exchange conditions and qualities of model data respectively building information.

Thus, the participants of the survey highly agreed (65%) to the statement that the quality of digital building models in form and content is not adequately standardized yet (Figure 1).

The specification of such defined process interfaces can be mentioned here as an important precondition for a simplified and secure contracting and cooperation: By referring to normative descriptions contracts can be concluded very efficiently and securely between client and planner respectively contractor as well as among the planners themselves. This becomes very important when the contract partners – like in Germany – are composed newly for each project.

On one side Germany's Architecture, Engineering and Construction (AEC) sector addresses this

problematic situation on a normative level by discussing a development of a set of standard specifications with regard to contents and process related quality criteria describing building models. On the other side an increasing number of larger construction companies and general contractors among practitioners have already started developing internal (process related) standards of quality. These are used as mandatory basis for co-operation towards project partners. Though the German authorities have also developed generalized quality guidelines, for instance the 'Dokumentationsrichtlinien für den deutschen Gebäudebestand' (documentation guidelines for German building stock (Figure 2)). These are capable of describing - in a BIM context - the execution of existing quality requirements in form and content. A main part of this directive is the alphanumerical building description that includes the constructional as well as the technical room data sheets. Thereby 'article' (countable types of furnishing respectively equipment that can be specified with further properties) and 'feature' (abstract definitions of objects' properties) are fundamental structures for description. Equal characteristics of plots, buildings, spaces, equipment etc. are described with a uniform code of property.

Uniqueness of quantity-on-hand data has to be permanently assured by all involved stakeholder in order to guarantee consistency towards forward projection with the streamlining exchange of digital building information data (coding of plots, buildings, floors, spaces and if necessary identification numbers) [1]. Thus corresponding concepts for process-accompanying quality assurance in the context of BIM were developed and exemplarily evaluated in pilot projects.

In the pilot project Humboldt-Forum (HUF) initiated by the Federal Office for Building and Regional Planning (BBR) quality criteria referring to this directive were specified. These criteria do not only regard to a future use of model data in facility management, e.g. space coding, topological allocations etc., furthermore they also cover important information for materialization and cost-estimation processes,

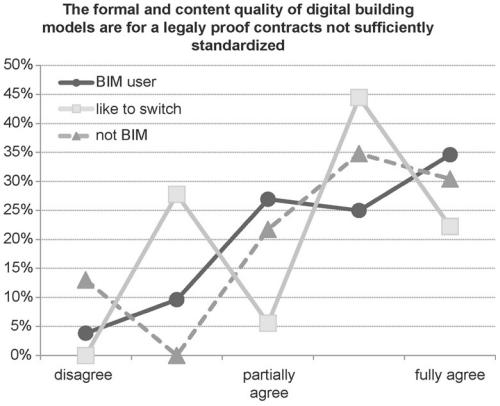


Figure 1
Agreement to the statement that the quality of digital building models in form and content is not adequate standardized yet.

e.g. part submission warrant and material coding.

Toward this quality standard a first step was taken by developing appropriate checking reports that enabled a process accompanying inspection and evaluation of model data received by project partners. Taking exchange scenarios into account, prior defined in the contract agreement, a process-accompanying quality control process was able to be executed. However, in the context of the HUF project manual (with eyes) quality control of the acquired alphanumeric criteria proved to be very extensive and ineffective. Thus a demand clearly emerged for suitable tools regarding model management and alphanumerical model analysis respectively model checking. The special need was to enable process-accompanying capabilities of generically rule-based semantic inspection of BIM models. After a market analysis it became obvious that so far none of the available tools had been able to satisfy the special demands of the authorities like a high level of flexibility and dynamic references. The business models

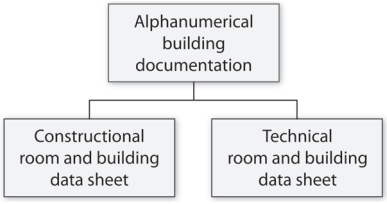


Figure 2
Detail from documentation guidelines for German building stock.

of the available tools did also not allow a flexible extension of the rule sets - they mostly only support a configuration of already existing basic rules. The implementation of totally new rules requires a charged development order. Thus, the BBR decided to develop own adapted BIM model checking software in cooperation with the KIT to extend their IT-infrastructure and meet their specific requirements.

REQUIREMENTS AND SOFTWARE CONCEPTS

Prefacing the description of the tool development, crucial requirements will be subsumed on the technical side and crucial aspects of the software conception are pointed out. Besides a brief overview on the technical implementation, the concept of rule configuration is subsequently presented more detailed.

On the technical level, the above derived logical requirements were further divided in those concerning the HUF context and the ones with regard to content. In terms of BBR – aspects related to actively running processes of the overarching BIM context in the ongoing HUF pilot project on the one side and on the other side the content- and organization-related approaches needed for implementation of automated compliance checks on documentation guidelines contents in process-accompanying BIM-model use cases with a long-term perspective on deriving robust internal IT standards results of the pilot projects. For BIM this meant the conception of a well reachable and compact workflow in order to organize the discipline-overarching data exchanges. Contrary, on the side of developing and managing generic rule repositories for documentation guidelines, flexible 'organizing'- functionalities stood alongside with requirements regarding the user's autarkic manipulation of rule logic in the focus of conception. Both levels of requirements were thematically separated and developed into two final versions of the software.

At first a clear conception guideline for development was given through the clearly described operative focus on one central workflow on the requirements side. This model checking workflow meant to be carried out by users without specific knowledge required - therefore developing aimed at a lightweight tool as a so called 'standard' version. In four steps it should be possible to load a model file, check its contents with chosen rules from a given repository and finalize a result report that can be exported as Excel-XML. While focused on the implementation in the HUF project it was meant to streamline first practical experience on BIM application by BBR.

The deduction of the development steps from requirements for the proper checking tool was confronted by an opposed development task – a rule configuration conception for BBR. A complex requirement level with emergent outcomes to be considered stood at the core of this task. For instance, technical implementation and maintenance of machine-readable logical contents did require designing totally new workflows (developing rule, maintaining repository etc.) accompanied by the need to define the new user roles (model checker, rule administrator). Setting up the overall conception had to be considered as a fuzzy context of future usage (after implementation in pilot project). Beside these 'invisible' requirements, future user/user's structures is unknown at the time of defining specific tool functionalities – all requirements for managing and manipulation of rule logic share common grounds e.g. need at different granularity of different previous knowledge on certain sets of circumstances (knowledge of model, knowledge of rules, knowledge of use case workflows etc.).These identified kinds of overarching concerns made it necessary to consider in the overall conception, the very diffuse yet universal requirements for the user interface logic of the software.

At last are the specific demands condensed in seemingly arbitrary requirements, for instance the reduction of inspection to only a part of the subject matter (IFC model instance), because of BBR's over time evolved structures. So while all rules concern the alpha-numeric part of IFC-model, this relates in terms of content to the requirements regard-

ing the rules to be checked with the software, as they deflect documentary guidelines with regards to content that were first adapted for BIM context and then transferred into generic rules for model checking. Regarding IT infrastructure, BBR already had software for checking model geometry (clash detection), which was not capable of providing sufficient user-specific configurable inspection of model semantics. In contrast to heavyweight/monolithic proprietary checking software that fail to support users with full autarkic rule development capabilities on behalf of large functionality range (geometry and semantics), causes a high level of complexity in terms of rule layout e.g. Solibri [2]. In a long-term perspective for instance, a central requirement was to keep this generic rule repository up to date by own means - in case of further developments in federal guidelines regarding buildings. Thus development aimed at a freely configurable and manageable user's rule base.

By especially enabling a separation of concerns between the knowledge of model users put into rule developments, and developers taking care of software related maintenance regarding the inspection shell, it/this seems to be the better business model as compared to mixing these concerns. Finally, the aim was integrated in the software conception by physically decoupling of the rule logic from the inspection-shell tool (business logic), and placing it in open described xml schemes describing the user's domain-specific rule logic.

IMPLEMENTATION AND USAGE CONCEPTS

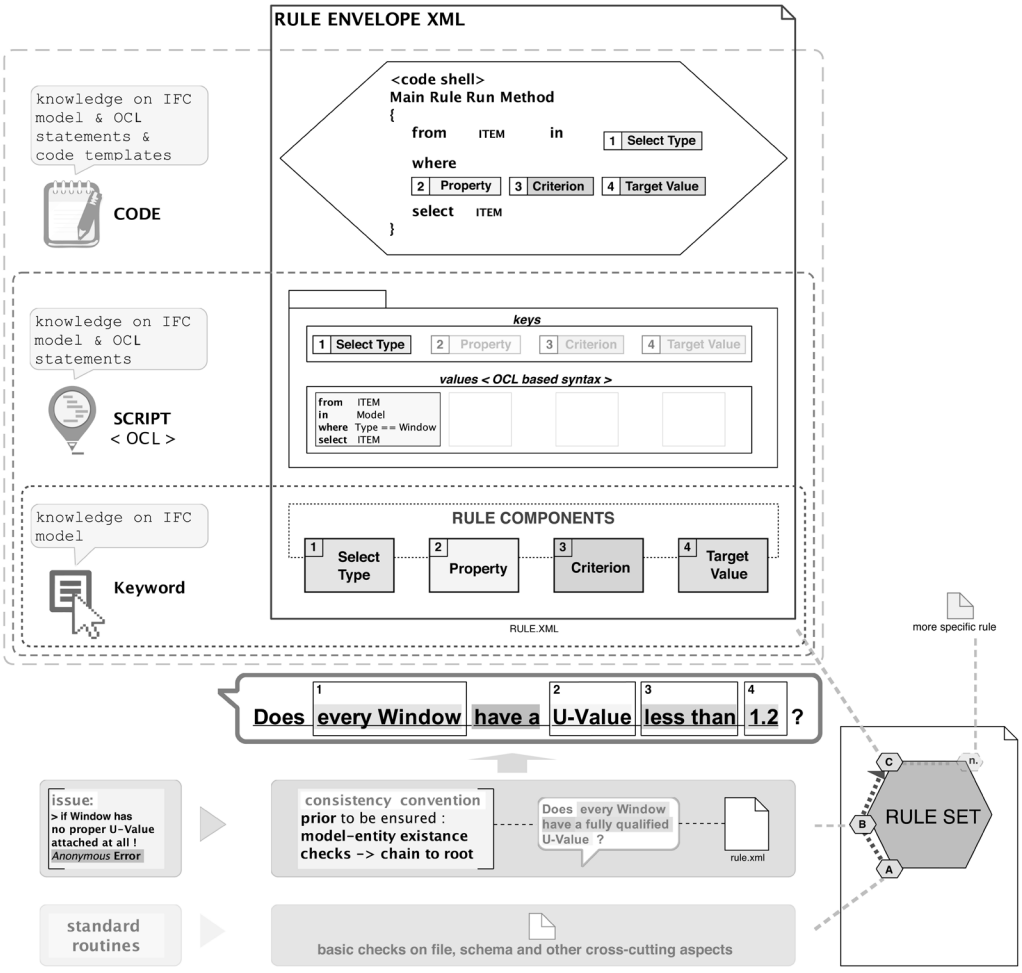
Besides serving BBR as a tool (standard version) in practice with ad hoc revenues within the HUF project, having all rule logic at hand (admin version) stays warranted if enhancements in rules take place because of acquired experience in practice over time that the user can successively put all together – self-tailored for internal workflows – into a common rule library and manage it from there.

Strategically it had been considered to implement the standard version as a subset of the extensive administration version. In this way, the rule development environment always has the model checking functionality at hand as needed for testing and verifying rule logic. Again placing the management of rules in the separate version generally provides a base for robust user's structure – e.g. versioning control separated from usage of rules.

In order to present the logical integration of the management environment *rule library* on a macro level e.g. the user-regarded convention-based frame settings, with the manipulation environment *rule configuration* on the micro level e.g. code, notations, standards, derived conventional settings, a bottom up approach will illustrate the latter level on behalf of the constitution of a rule respectively crucial aspects into turning logic to automated compliance checks e.g. as a cross cutting concern of the convention on how one rule logic expands over several rule files.

On the bottom level, the logical counterparts to the main XML tags in a rule file refer to loosely coupled constitutes of one logical clause – a so called *rule envelope*. All assigned rules with regard to contents were therefore decomposed in rule components that serve as elements in a model kit – embodied in the *rule configuration*. An overriding ‚orchestrating' algorithm, that is distinct according to the issue of the rule logic and sets the logical parts of the specified rule clause in relation to each, other respectively condenses the parts to the above introduced serial process-able check workflow. As in every rule file components are linked likewise to a checking workflow on the level of the beneath 'code skeleton' (cloze with wildcards for the return values of the rule components), a set of core templates were introduced as basic rule envelopes. Every variegated rule is derived from one kind of query at the core - in other words this origin (basic rule) connects all other sibling rules in the tool to each other. Allocating one basic approach to each general central questionings gives the user a decomposition principal at hand for formation respectively further development of the own rule repository and eases editorial aiding of a current state through elimination of redundancies.

Figure 3
Software conception: logical contents of rule XML.



Along the basic structure of an example rule (Figure 3), the rule components will be introduced. The decomposition of the components, hence the basic structure was thereby adopted from the Object Constraints Language's (OCL) main statements [3] in order to promote a convention-driven limitation to the source code used in the user's rule files by only implementing common templates.

Regardless of the user's knowledge, a verbally represented questioning (in the example: "Does every window have a U-value less than 1.2?") can function as an initial point of the rule development. In a first step, the rule administrator dismembers this concrete question according to the four components of the check workflow. Depending on a basic population of all components embedded in the

rules of the repository, it can at this stage already be possible to suit the logical clause of the rule, only by choosing existing instances of components and if applicable alter the specific values.

Specific quantities of element instances are already gathered during loading the model in order to show a brief summary on the model content. These can also be adapted in the components by referring to their specific keyword. In cases where specific quantities are not set up at start nor defined in another rule's context, the rule administrator alter a similar query or create a new one. Thus specified in the so called Select Type component, the related congregation of instances is being hooked to a main iteration mechanism/slope in order to check a circumstance on each of the selected IFC Type instance.

Further specification of a property to the prior chosen IFC type is defined in the so called Property component (in the example: the pointer to the U-Value in the property set of the window type). So far the specified components will allow the main iteration through specific actual values of the selected congregation - in order to check these values, a corresponding desired output of the value is assigned in the so called Desired Out/ Target Value component. By defining an appropriate condition for the relationship between actual and targeted value in the so called Check Criterion/ Criteria, the last part of describing the rule clause is accomplished. By standard, this now fully specified checking workflow will return all elements of the congregation that do not meet the specified condition – a subset with error prone instances. In order to achieve a comprehensive output of the checking results, the so called Result component can be adjusted by specifying a suitable error description as a cloze with wildcards for crucial single values to clearly outline the returned error prone circumstance. With access to functions of standard version for model checking and functionality for verifying of developed rule, the rule configuration is set up with all necessary means of editing. This is streamlined by a concept of usage (UI) which is decomposed in different levels aligning with required stages of user's knowledge.

At the structural base of the hereinafter described concept of the knowledge-oriented usage strategy for dealing with complexity of involved matters, stands again the decomposition into the above mentioned rule components. Three access levels (in Figure 3: dashed lined boxes) are provided to the user in the rule configuration. In every higher level thereby – as a further abstraction of complexity - less knowledge is required to manipulate rules with regard to contents. However, this abstraction is again streamlined by limited amount of editing possibilities. Only elements that have been previously defined in the level below can therefore be referred to in the higher level. At the top level, keywords enable the cascading/coverings of complex logic to the unseen background. On this level it is possible to create variants of existing rules only with combination of given components and knowledge of the rule constitution. When additional object quantities are needed, domain knowledge of the constitution of the model to be checked is already crucially required – the only further knowledge required is on how to handle the statements, as described by the OCL standard.

In a middle level queries are therefore described in OCL-conform syntax and made referable by a self-chosen synonym (keyword for the top level), together they are persisted in the XML rule file as key-value-pairs. Knowledge to its full extent – rule constitution, domain-related as well as application of the code skeleton being used (basic rule template) – are only crucial on the ground level in order to create or extend the basic rules. With this concept at hand, the BBR staff is on the one hand in the position to ad hoc address different questionings within the HUF project on a base of manipulating the rolled out basic rules, and they are on the other hand in a long-term perspective by gathering necessary expertise over time, so that the user can then be enabled to maintain and further develop rule repository on own means.

Through inclusion of the configuration environment with the rule repository in the hereinafter described rule library environment, an overarching

management of the development work on base of the rules is ensured. Bound by convention - all rules are obligatory referred to by at least one rule set – in the rule library environment the user is provided with manipulation functionalities of rule sets only. By the ability of individual assignment of rules to respectively different sets, a user-specific structuring is also enabled e.g. project-oriented. The structuring of the own rule repository is on the technical side supported by a technology known from common Integrated development environments (IDE) for dynamically compiling and executing rule code during runtime of the tool. Thus this enabled physically decoupling the user-specific logic concerning knowledge on model domain, rule repository, workflows etc. from the checking shell, since the current repository is loaded from the XML database at runtime only. Therefore, the own rules can also be structured and managed independently of the tool, if for instance there is available infrastructure like Content Management System (CMS) etc.

A basic structuring along the domains of the checked models was loosely integrated by strictly keeping XML parsing separated from checking mechanism in business logic and user-domain logic. Since the tool initially construed for checking the IFC XML standard already at core of basic rule development, a latter decoupling of basic XML-operations from the domain-specific aspects was focused. The basic rules that were therefore rolled out with the software, give the user a solid base to start an own robust repository from. Thus in terms of different future demands, it could also be further developed in a model-overarching manner.

As pointed out in different aspects covered above, making the software more flexible on the technical side is crucially accompanied by transferring of logical and structural responsibility to the user as well. This means for instance that before checking specific details, all model-instances involved have to be checked whether their instances are completely implemented in the way the specific rule refers to them. Since a non existing instance of a model element will not be recognized as error prone because of the fact that it is non existent, it is not covered by the questioning concerning its value check. In this case, the user has to warranty that inside the set of the specific rule, there is a prior rule that checks the existence of covered elements before the more specific check on the element value takes place. Since there are "chains" of existence checks for every specific rule in order to ensure the overall correctness of covered contents - all involved elements - there seems to be great potential for internally standardizing this issue and structuring it as a basic subset in the repository. Although turning the exhaustive responsibility for the domain-specific logic in the user's hand is accompanied by complexity in terms of managing the rule repository in the case of BBR application this seemed reasonable. Beginnings in the field of BIM model check research can be seen especially in countries were BIM-models are put to practice, for instance Balaban (2012) supporting Turkish authorities with automated compliance checks for fire guidelines. However segmented chains of requirements due to complex application contexts seem a common ground thereby. In order to achieve the goals mainly focuses get carried on very specific context-dependent solutions and therefor often show contradictive mixing of user and developer concerns. Hence they seem to be more vulnerable in a long term perspective; it is difficult to find a suitable allocation of developer and user respectively a suitable organization of the further development of the rule repository after solutions are implemented.

Because the pilot character of the HUF project in the usage of BBR, it was necessary to create a totally new rule repository. The separation of concerns was suitable since the user first has to build up expertise regarding different domain-specific disciplines. It enables thereby also successively gaining grip on the full functionality range of the software.

Other than these concerns regarding logical and structural responsibilities, all overarching concerns regarding plainly information technology were considered in the tool shell. Thus throughout all developing processes, a strict internal library-wide

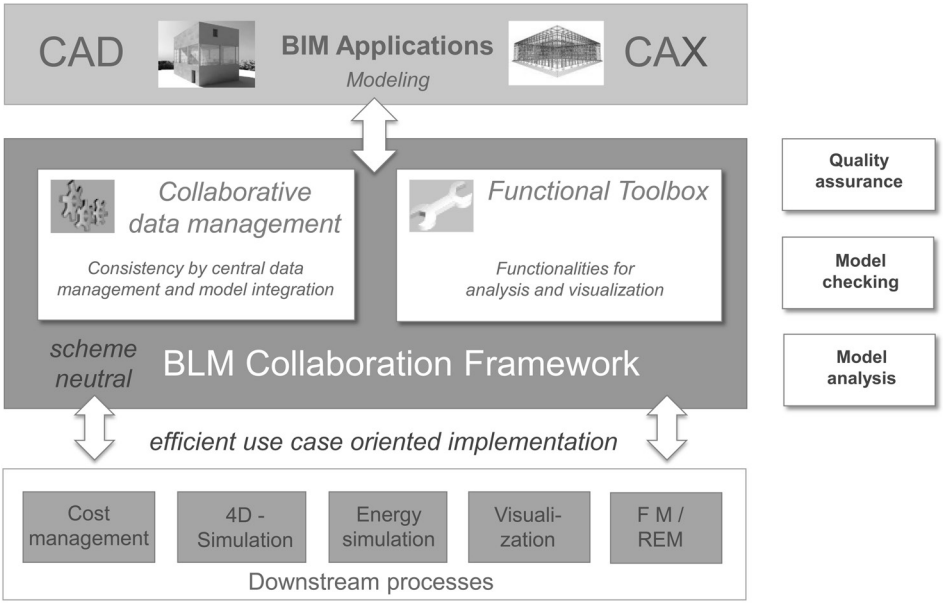


Figure 4
Concept of collaboration framework.

versioning keeps every applied rule with regards to content referable to every produced inspection result. This seems important e.g. it guarantees essential consistency for archiving purposes – older inspection reports are always mapped to the valid rule at time of inspection. Import and export functionality enables an administrating user to deploy newly finalized rules to users of a standard version that only use production ready rule sets. This allows team-internal organization of roles within a user group. Together, versioning and a project specific structuring enables parallel usage-oriented organization of similar rules in different contexts.

Putting ModelCheck into practice (HUF project) enables iterative evaluation and optimization regarding rule logic for the user – whereon further development of inspection shell functionality is then based upon. ModelChecks further development also takes advantage of extensions to the underlying framework, which will be briefly introduced hereinafter.

TECHNICAL BASE OF IMPLEMENTATION

As an application-independent central service, the framework provides base functionalities for the model based data analysis. The base functionalities serve collaborative data handling, e.g. type- and attribute-oriented selections of partial and aspect models, integration of these partial models as well as supplying mechanisms for versioning, change management and transaction control (Figure 4).

Thus a kind of "Meta-Model-Server" is provided for further research and development projects that in different application scenarios can be implemented for different kinds of model standards like ifc STEP, CityGML or GAEB. Furthermore it supports model overarching model-operations (Hartmann and von Both, 2011).

OUTLOOK

Further development on the analysis and visualization components in the context of energy efficiency will take place within the science project "EneffBIM"

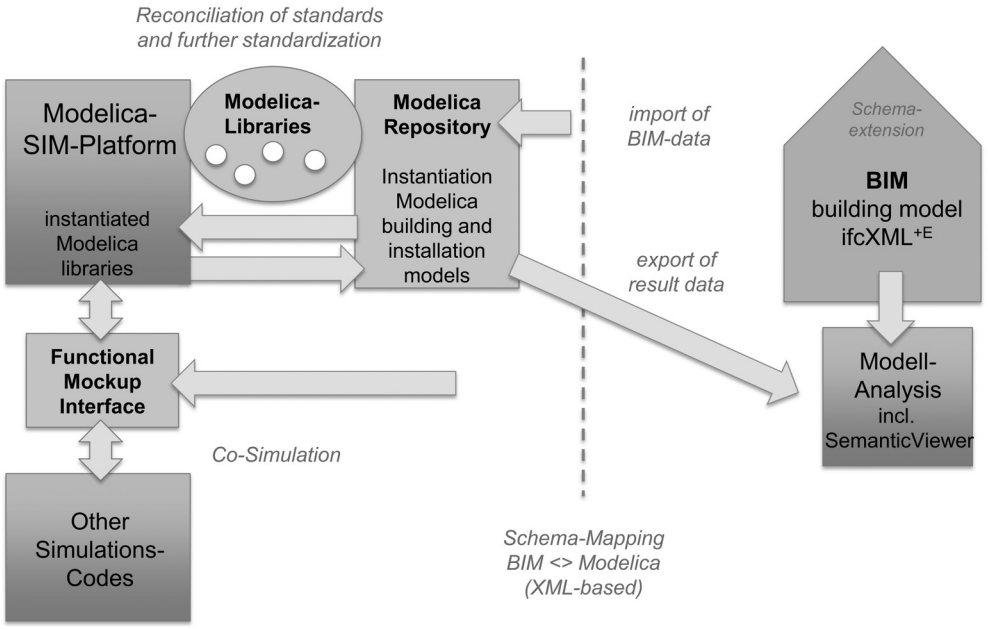


Figure 5
Conception draft EneffBIM project.

(funded by the German funding program "EnTools" released by the German Ministry of Economy) starting summer 2013. As seen in Figure 5, especially for the usage of dynamic energetic simulation, the logical content-regarding model analysis shall be the quality management vehicle for securing the interface from BIM to simulation.

With involvement of different Frauenhofer institutes (ISE and IBP) as well as the universities RWTH Aachen, UDK Berlin, KIT and buildingSMART on one side, the IFC model will be extended with energy relevant base types (input parameter) and suitable geometric representation forms. On the other side regarding energetic simulation, tools for co-simulation in the context of Modelica will be further developed and a synchronization of existing model libraries is been focused on.

Concerning model checking aggregated simulation results shall be led back in the BIM model in order to ensure better re-transition and evaluation of simulation results into the planning process and towards the layer of decision making. ModelCheck then serves as a checking and analysis tool for evaluation of variations with their different simulation results.

In this case a great meaning is beard to semantic visualization of simulation results (specific constraints of property values), representation of the range in values that exists in the comprehensive model and also checks on the characteristics of values.

REFERENCES

von Both, P. 2012: Potentials and Barriers for Implementing BIM in the German AEC Market - Results of a current Market Analysis, contribution to the30th eCAADe conference, Prague, 09/2012

Hartmann, U.; von Both, P. 2011: Ein Framework zur Definition und Durchführung interdisziplinärer, modellübergreifender Analysen am Beispiel solarer Einstrahlpotentiale im urbanen Kontext. BauSIM2012 Conference - Gebäudesimulation auf den

Größenskalen Bauteil, Raum, Gebäude, Stadt, Stadtquartier. Berlin 09/2012.

Balaban, Ö. 2012: Automated Code Compliance Checking Model for Fire Egress, contribution to the30th eCAADe conference, Prague, 09/2012

[1] http://www.bfr-gbestand.de/inhalt/inhalt.html.

[2] http://www.solibri.com/

[3] https://en.wikipedia.org/wiki/Object_Constraint_Language