In Search of Open and Practical Language-Driven BIM-Based Automated Rule Checking Systems

69

Wawan Solihin, Johannes Dimyadi, and Yong-Cheol Lee

Abstract

Significant progress has been made towards BIM-based automated rule-checking systems. There are multiple approaches that show varying potentials as crucial components for open and practical rule checking systems. However, in the current state, we are not yet quite there as they are still several barriers that prevent the needed shift from proof-of-concept to the real-world implementation. This paper reviews various language-based rule checking systems that have been proposed and assesses their potentials and gaps that need to be overcome for them to become practical. It introduces metrics of eleven criteria to analyze various approaches to assess their readiness for the real-world implementations. The criteria cover a wide range of aspects including language expressiveness, ease of use to define a rule, openness, level of maturity, and performance. These criteria help to identify gaps that currently exist that need to be overcome to allow a leap from a proof-of-concept to the real-world implementation. From the assessment, it is obvious that no one single approach is currently capable of covering the entire spectrum of requirements for automated rule-checking systems. The assessment also shows that the possibilities of combining two or more approaches may accelerate the realization of an open and practical language-driven automated rule checking system.

Keywords

BIM • Automated rule checking • BIMRL • Language-driven rules

69.1 Introduction

The benefits of automated rule-checking systems have been recognized. However, the full benefits of such systems have not been fully realized. It is because of multiple challenges that range from the industry readiness with BIM, a wide-range acceptance of the standards, to the availability of the suitable tool to achieve the purpose. In recent years, many factors that inhibited the practicality of automated rule checking systems (RCS) have been overcome. The industry has been adopting BIM as part of their business practices, IFC standards have become mature enough, many BIM authoring tools are supporting the standards with increasing quality, and several countries have started mandating BIM requirements for the regulatory submission such as Singapore and the UK [1]. Despite all of these, the progress on the automated rule checking systems is still confined largely within the research domain [2]. More recently though, there have been renewed interests to revisit the real-world implementation of automated rule checking systems. This can be seen with such initiatives as the Korean government sponsored KBim project [3], the BuildingSMART Regulatory Room working group to identify

W. Solihin (✉)
novaCITYNETS Pte. Ltd., Singapore, Singapore
e-mail: wawan.solihin@gatech.edu; wawans@nova-hub.com

J. Dimyadi
The University of Auckland, Auckland, New Zealand

Y.-C. Lee
Louisiana State University, Baton Rouge, LA, USA

I. Mutis and T. Hartmann (eds.), *Advances in Informatics and Computing in Civil and Construction Engineering*,
https://doi.org/10.1007/978-3-030-00220-6_69

potentially commercialisable rule checking systems [4], and the most recent RFPs from the Singapore government agencies: Building Construction Authority's (BCA) iGrant, Urban Redevelopment Authority's (URA) Intelligent Code Checking System, Public Utilities Board's (PUB) BIM Checking System. The current issue is that such system still requires high cost to develop. It is our interests to find an open and practical approach towards the system that allows much lower barrier to entry for such systems.

From the published research papers, there are several approaches that have been proposed to solve the challenges of the rule checking systems. At least two ways to classify different approaches have been proposed based on the complexity of implementation [5] and based on the areas of problems they address [6]. No one approach can cover the entire range of components required by a rule checking system. They vary in term of their complexity, performance and the required domain knowledge to define and run the rules. Since a rule checking system requires a way to define rules, it is logical that such a system will be language based. It provides flexibility to deal with wide range of rules and their requirements. This paper focuses on language-based approaches, even though some tools that limit language expressions with their user interface are also considered.

69.2 Review of Language-Based Rule Checking System (RCS)

There is a wide range of rules applicable to building models. Given the extent of rules involved, it is difficult for a single application to provide a general solution to the problem. The language-based approach, if defined well, potentially will provide a platform to allow basic building blocks that can be used to assemble much more complex and varying rules. The richness of the language determines the ease of defining rules but also sacrifice on the generality of the language for less complex and compact rule definitions. In the software world, this type of language is known as Domain Specific Language (DSL). The DSL is an efficient way to deal with specific domain requirements as a mean of making the language expression compact and efficient to address domain requirements. Rule languages for BIM fall generally into the DSL category because it is specific to building information and rules pertaining to buildings. There are two genres of DSL, i.e. an internal DSL and external DSL. The internal DSL "borrows" an existing language as the host language, extend it and adapt it to create a feeling of a particular language. Included in this category are: Lua script used in a commercial application called FORNAX™ to program rules using its C++ based APIs [7], SWRL (Semantic Web Rule Language) that uses series semantic web triplet concept to define a rule [8], SPIN (SPARQL Inferencing Notation) that combines concepts from object-oriented languages, query languages and rule-based systems to define rule and constraint languages that is based on the Semantic Web, and rule-based system such as DROOLS. XML based rule languages, such as mvdXML [9], RuleML, LegalRuleML [10], BPMN [11] also fall into the same category. They have advantages of being part of a larger community and standardized languages that provide a wider range of available tools and resources. However, since the languages may not be designed specifically for BIM, there will be a need to work around, extend or twist the language to fit BIM needs. This often leads to an awkward syntax and extra steps. The external DSL, on the other hand, is usually more concise or compact and feels more natural in defining the BIM-based rules since it is specifically designed and tuned for them. The disadvantage is that it requires its own custom parser and a new syntax that user needs to learn. Languages that fall into the external DSL include QL4BIM [12], BERA [13], KBim that uses meta-programming concept [14], and the authors' own BIM Rule Language (BIMRL) that combines SQL based query language, built-in support for spatial operators and function extension [15]. External DSL also includes visual language such as Dynamo on Revit, Grasshopper on Rhino, Marionette on Vector Works, and VCCL [16].

In addition to the general categories of internal and external DSL, there are other tools that are rather difficult to fit into the above, for example, NLP [17] that uses AI techniques to analyze building rules directly into executable rules, and RASE which uses manual markup into the rule texts and transform the rules into IFC constraint model [18].

With so many proposed approaches, it is difficult to have an overview of how much each approach and tool can satisfy BIM rule checking requirement. The initiative by the building SMART's Regulatory Room to gather such information and assess potential commercial applications of such tools or languages have not identified any tools that are sufficiently complete and ready for the real-world usage [4]. To address this gap, the authors look into various factors that are important in assessing the suitability or readiness of such tools for the potential real-world application. Based on these factors, the metrics with eleven criteria are proposed.

69.3 The Eleven-Criteria Metrics Assessment

The authors develop a set of criteria to help place those approaches and indicate their readiness towards the real-world implementations. The criteria are developed based on the need to have an open standard for rule language, which is capable to deal with complex rule requirements, concepts that fits for real-world implementation, ease of use. These arise from a combination of the real experience developing real-world rule checking system, works in the OpenBIM domain, and research works on RCS. Each of the language or approach will be assigned one of five scores: Low (◯), Moderate (◑), High (●), Not Applicable (▬), or Unknown (◈). While it is not always easy or possible to compare all the languages since there are no common examples for one-to-one comparisons, the authors try their best to use available references or examples to determine the appropriate classifications for each of the languages. One of the sources is from the building SMART's Regulatory Room's project on the automated rule checking, where several of the languages are applied to one building code based on a piece of regulation (article 34.1, Fig. 69.1) in the Korean Building Act providing a good source of comparisons even though not all are presented in a complete form for the rule [4]. It is not the aim of this classification to identify which language is better compares to the others but to show potentials and gaps that will be useful in order to help shift the current status towards a real-world implementation. Using this assessment, the authors hope to encourage further development of the language or to develop a combined research between two or more approaches for one or more practical implementations that the industry could benefit in the immediate future. The assessment is shown in Table 69.1.

The criteria include the following metrics:

1. The formal or standard schema for the rule definition

The existence of standard schema for rule definition is one of the important aspects of openness and standard way to represent a computable form of rules. While there are several rule languages that can be used to define rules in standard computable forms, only LegalRuleML, mvdXML, and RASE that qualified on this measure because they define a standard schema and specific to BIM.

2. Language expressiveness

Language expressiveness is important to assess how capable the language is to define wide-range of rules. Less expressive language will be difficult to use and may become too clumsy when used with complex requirements typically found in building codes. BIMRL that is specifically designed to deal with BIM related rules with minimum programming knowledge needed is highly expressive and compact. The example of BIMRL (partial snippet) for the Korean Building Act is given in Fig. 69.2.

Korean Building Act 34.1 ***"Installation of Direct Stairs"***

On each floor of a building, direct stairs leading to the shelter floor or the ground (including slope ways; hereinafter the same shall apply) other than the shelter floor (referring to a floor having a doorway leading directly to the ground and the shelter safety zone of a skyscraper under paragraphs (3) and (4); hereinafter the same shall apply) shall be installed in the way that the walking distance from each part of the living room to the stairs (referring to the stair nearest to the living room) is not more than 30 meters: Provided, That in cases of a building of which main structural part (excluding a performance hall, assembly hall, auditorium and exhibition hall which are installed on underground floors and which have a total floor area of not less than 300 square meters) is made of a fireproof structure or non-combustible materials, the walking distance of not more than 50 meters (in cases of multi-unit dwellings higher than 16 storeys, not more than 40 meters) is permitted, and in cases of a factory prescribed by Ordinance of the Ministry of Land, Infrastructure and Transport, which is equipped with automatic fire suppression systems such as sprinklers, in an automated production facility, the walking distance of not more than 75 meters (in cases of unmanned factories, 100 meters) is permitted.

Fig. 69.1 The Korean Building Act 34.1 "Installation of Direct Stairs" [4]

Table 69.1 The eleven-criteria metrics assessment

Language or tool **Legend** ○ Low ◑ Moderate ● High — Not Applicable ◈ Unknown	1. Formal or standard schema for the rule definition	2. Language expressiveness	3. Ease of defining rules	4. Minimum logic or programming constructs	5. Minimum domain knowledge required	6. Support for complex rules	7. Integrated geometry engine	8. Performance indication	9. Openness	10. Interface to other languages and systems	11. Level of maturity
NLP	—	—	—	○	○	○	—	◈	○	○	○
RASE and IFC Constraint Model	○	◑	◑	◑	○	○	—	◈	●	○	○
LegalDocML and LegalRuleML	●	◑	◑	◑	○	○	—	—	●	○	◑
BPMN	—	◑	◑	○	○	○	—	—	●	—	●
Mvdxml + IfcDoc	●	◑	◑	●	○	○	—	◑	●	○	◑
BIM Assure	—	◑	●	●	●	○	—	●	○	○	◑
QL4BIM	—	◑	◑	◑	○	○	◑	◑	●	○	○
BERA + BOM	—	◑	◑	◑	◑	◑	○	◑	◑	○	○
<u>KBim</u>	—	◑	○	○	○	◑	○	◈	○	○	○
BIMRL	—	●	●	●	●	◑	◑	●	●	◑	◑
DROOLS	—	◑	◑	◑	○	○	—	◑	◑	●	●
Rule Table	—	○	●	●	○	○	—	◈	◑	○	●
Prolog	—	○	○	○	○	◑	—	◈	◈	○	◑
SWRL + IfcOWL	—	◑	○	○	○	○	—	◑	●	◑	●
SPARQL and SPIN + IfcOWL	—	◑	○	○	○	○	—	◑	●	◑	●
Revit Model Review	—	○	◑	●	○	○	—	◑	○	○	◑
Dynamo + Revit	—	○	●	◑	○	◑	◑	◑	◑	○	●
Grasshopper + Rhino	—	○	●	◑	○	◑	◑	◑	○	○	●
Marionette + VectorWorks	—	○	●	◑	○	◑	◑	◑	○	○	●
VCCL	—	○	◑	○	○	○	—	◈	○	○	○
FORNAX Lua scripting	—	○	○	○	◑	●	●	●	○	○	●
[a] Solibri Model Checker (SMC)	—	—	—	—	◑	◑	●	●	—	—	●

[a] *SMC is included here due its popularity even though it is not strictly language-based rule checking system*

3. The level of ease to define rules

A language should be ideally easy and intuitive to define building rules. The different tool will have a different level of difficulty. While a certain level of complexity will be expected for complex rules, it helps if the language is consistent no matter whether it is a simple rule or a complex one. An example that allows easy definition of rules is given below for a commercial application BIM Assure that is UI driven. This makes it very easy for typical users to define any rules, albeit restricted to certain types of rules that work on explicit properties and relationships. In this category, most of UI-driven language score well, but not all are given full High mark due to their generic language styles that require a lot more programming type of definition to achieve the same impact.

```
CHECK
{ ifcspace s, ifcdoor d
   WHERE property(s,HabitableSpace)='TRUE'
and classificationof(d).classificationitemcode='2.6.2'
and classificationof(d).classificationname='BCIS'
and s.container=d.container
   COLLECT s.elementid spaceid, d.elementid doorid, s.name spacenumber;
} AS SET1;
{ IfcSpace SC
   Where CONTAINS(SC,USEGEOMETRY).ElementType='IFCFLOWTERMINAL'
and CONTAINS(SC,USEGEOMETRY).Name like 'Sprinkler%'
   COLLECT SC.ElementId SPACEID, SC.Name SPACENUMBER,
SC.LongName SPACELNAME, COUNT(unique CO.ElementID) SPRINKLERNO
   GROUP BY SC.ELementId, SC.Name, SC.LongName
} AS SET2;
EVALUATE computePathAndDistance(spaceid, doorid) output ?traveldistance
From SET1 LEFT OUTER JOIN SET2 USING (SPACEID, SPACENUMBER);
ACTION
   WHEN ?traveldistance>30000 and
NOT (?BuildingClassification='En_30_50_50'
and (?SprinklerProtectedAuto ='yes'
OR ?ProtBySprinklerSystem>0
OR SPRINKLERNO>100))
OR NOT ((?NonCombustibleCnt/?MainStructCnt)>0.8
OR ?FireProtectionClass='FireProof')
   { … }
```

Fig. 69.2 A rule snippet in BIMRL for the Korean Building Act 34.1 [4]

```
rule "KOREAN-0-1-0-OUTSCOPE" salience 75 when
   exists Applicability(id=="KOREAN-0-1",applicable==true)
   not Building(fireproof==true||made_of_fireproof_materials==true) from
bim.fetch("Building")
   not Building(fireproof==null || made_of_fireproof_materials==null) from
bim.fetch("Building")
   then results.na("KOREAN-0-1-0"); end

rule "KOREAN-0-1-0-FAIL" salience 0 when
   Applicability(id=="KOREAN-0-1",applicable==true)
   not Result(id=="KOREAN-0-1-0")
   exists Floor(walking_distance_from_living_room_to_stairs>50) from
bim.fetch("Floor")
   not Floor(walking_distance_from_living_room_to_stairs==null) from
bim.fetch("Floor")
   then results.fail("KOREAN-0-1-0"); end
```

Fig. 69.3 A rule for the Korean Building Act 34.1 written in DROOLS [4]

4. Little or low requirements for logic or programming constructs

Due to the wide range of complexity of BIM rules, it is inevitable that some level of programming or logic will be needed to encapsulate complex rule requirements. These criteria assess the level of dependency on programming constructs. Generally, the lower level of language allow flexibility to write complex rules but at the expense of general accessibility to non-programmers. Higher level language will be easier to be understood, but usually hard to deal with complex rules. An example of such rule written using DROOLS rule engine is given in Fig. 69.3. Combination of the two may be desirable to address this issue, but the language must have built-in capability to support this. The possibility of language extension such as through plugin mechanism may provide the much-needed support. Tools such as FORNAX™ Lua based script has been shown to be able to deal with complex building codes, but it requires specialized skill in Lua programming to do this.

```
<Rules>
  <AttributeRule RuleID="PredefinedType" AttributeName="PredefinedType" />
    <AttributeRule AttributeName="IsNestedBy">
      <EntityRules>
        <EntityRule EntityName="IfcRelNests">
          <AttributeRules>
            <AttributeRule AttributeName="RelatedObjects">
        <EntityRules>
          <EntityRule EntityName="IfcDistributionPort">
            <AttributeRules>
              <AttributeRule RuleID="Name" Description="The name of the port."
AttributeName="Name">
         <EntityRules>
           <EntityRule EntityName="IfcLabel" />
         </EntityRules>
           </AttributeRule>
             <AttributeRule  RuleID="Flow"  Description="The  flow  direction  of
the port." AttributeName="FlowDirection">
         <EntityRules>
           <EntityRule EntityName="IfcFlowDirectionEnum" />
         </EntityRules>
           </AttributeRule>
            <AttributeRule   RuleID="Type"   AttributeName="SystemType   <Enti-
tyRules>
```

Fig. 69.4 An example of the mvdXML rule [9]

5. Little or low level of domain knowledge required

This assesses how much expertise and specific domain knowledge a user would need to be able to use the language or system to define rules. Since the rule is BIM specific, most of the languages require intimate knowledge of building domain and building codes. The Medium rating is given only to those that simplify IFC. Only BIM Assure that provides pre-mapped categories and UI guided rule definition. Figure 69.4 shows part of the mvdXML rule. Since it is tightly linked to IFC, the intimate knowledge of IFC is a must. All rules using mvdXML operate within IFC model and its entities.

6. Support for complex rules

Complex rules typically involve geometry and spatial operations. In this measure, most of the languages are given Low since they mostly do not support integrated geometry and spatially related rules. Only FORNAX™ with its comprehensive API with integrated support of geometry and spatial operations is given a High rating. The example of Lua scripting in FORNAX™ is given in Fig. 69.5. The visual programming tools are given only Medium because they do not have direct support for spatial operations and higher semantics knowledge required in complex building codes. Most of the other languages will require programming extension to support such requirements.

7. Integrated geometry/spatial engine

In many BIM-based rules, there are many rules that require geometry and spatial reasoning to perform rule checks. Therefore, it is critical to have access to such geometry engine. Ideally, this should be fully integrated that allows a single coherent language can be used to perform queries and check in one integrated manner. This topic has not been very well researched, probably because it involves rather complex requirements when dealing with geometry. Only QL4BIM and BIMRL are known to consider spatial queries to be integrated into the language. Product specific visual language such as Dynamo, Grasshopper, and Marionette are capable to work on geometry too, but not a complete set of spatial operations.

8. Performance indication

What is the performance indication of the system to execute various rules? The authors are unable to collect all the necessary performance indicator of some of the languages. They are left with a question mark. In this measure commercial languages such as FORNAX™ Lua and BIM Assure perform best as they are designed to perform. The only one that is also designed for efficient query and checking is BIMRL.

```
function checkfunction(storey) -- common codes for GFA function
   FXGFA.DebugMsg("__| BuildingStorey ".. storey:GetAttri("Name") .." |__")
   ParseObjectGroup(storey)
   local elevationStorey = storey:Elevation();
   local grpCoverElement = FXGroup.new()
   for key, value in pairs(tblCoverElement) do
      grpCoverElement = grpCoverElement + value
   end
   local grpElementType = FXGroup.new()
   for key,value in pairs(tblElementType) do
      grpElementType = grpElementType + value
   end
--[[GFA 7.30.5 Gondola platform at roof top]]
   local  FinalPrj  =  FXGFA.GondolaPlatformAtRoofTop(storey,  elevationStorey,
grpCoverElement, grpElementType);
   if isIncluded == true then
      FXGFA.IntegrateGFA(storey, FinalPrj, elevationStorey, msg)
   else
      FXGFA.IntegrateNonGFA(storey, FinalPrj, elevationStorey, msg)
   end
end
```

Fig. 69.5 FORNAX™ Lua scripting example for gross floor area computations

9. Openness

The issue of openness of the language or tool is an important factor that allows utilization of available tools, ready knowledge available, and some assurance for longevity because the specifications are open.

10. Interface to other languages and systems

Ability to interface to other languages and systems will be crucial to enable much more complex or complete solution for the building rules from the definition, to the runtime of the rule execution. This also helps to complement the tool with other tools that may be addressing the more specific problem instead of a single or monolithic approach to the BIM rules. Open sourced rule engine such as DROOLS allows interfaces to other Java-based tools.

11. Level of maturity

The level of maturity is measured by how long the language has been introduced, how widely it is used, how actively it is maintained, and how many implementations that have been done with it. which include how recent the language or system is developed or maintained, whether it has been adopted into a real system, and the scope.

69.4 Conclusions

The ideal language should be rated high in all the criteria. Based on the assessment, none of the languages fulfill that. There are several promising languages or systems have emerged such as BIMRL, DROOLS rule engine, and on the commercial side FORNAX™ Lua scripting, which has the most mature implementation and supports complex rules. They still need to have the ability to support standard rule specifications. This may be achieved by supporting specification language such as LegalRuleML. The visual programming languages also show a lot of potentials, even though currently are mainly supported within proprietary native systems. Adoption of the visual programming language into already strong implementations may offer very promising user-friendly RCS.

The widest gap seems to be in the area of geometry and spatial support, and integration with other systems. Most languages treat the rule specifications, rule execution, and the data to be distinct and in a neatly separated compartment and hence hardly any that integrate the geometry and spatial into the language. This assumption is largely limiting since in many building codes the rule specifications often interdependent with the logic of checking and the data. BIMRL and QL4BIM offer such support, but to be able to support truly complex rules, they may need to be augmented with procedural languages such as C++, C#, Lua, or Java. BIMRL is designed to work with plugin concept and integration of the plugin into the language as an extension in a similar manner to a stored procedure in the relational database world.

The authors hope that this study helps to identify the priority and a more focused approach for BIM-based rule development and by not reinventing the wheels. By extending the already promising language and combining with other language(s), the result will be closer to the ideal rule language for BIM and therefore making the BIM automated RCS closer to its practical availability and its full potential to help the AECO industry. To bring BIMRL closer to the ideal language, we intend to enhance it by incorporating a visual programming language, a formal rule specification such as LegalRuleML, and possibly with a more powerful engine such as FORNAX™ Lua.

References

1. BCA: All set for 2015: The BIM Roadmap. Build Smart Magazine, no. 9, p. 4 (2011)
2. Dimyadi, J., Amor, R.: Automated building code compliance checking—where is it at? CIB WBC **2013**, 172–185 (2013)
3. Kim, I.: Automated building code checking related activities in Korea (2015). Online at http://iug.buildingsmart.org/resources/London/Regulatory%20Room/korea_automated-building-code-checking-related-activities-in-korea_khu
4. bSI Regulatory Room WG: Report on open standards for regulations, requirements and recommendations content (2017)
5. Solihin, W., Eastman, C.: Classification of rules for automated BIM rule checking development. Autom. Constr. **53**, 69–82 (2015)
6. Dimyadi, J., Solihin, W.: Editorial: CIB W78 special track on compliance checking. J. Inf. Technol. Constr. **21**, 315–316 (2016)
7. NovaCityNets: FORNAX (2016). http://www.novacitynets.com/fornax/index.htm
8. W3C: SWRL: a semantic web rule language—combining OWL and RuleML. W3C (2004). Online at http://www.w3.org/Submission/SWRL/
9. Chipman, L.T., Wiese, T.: mvdXML, Build. Model Support Gr (2013). no http://www.buildingsmart-org/downloads/accompanying-documents/formats/mvdxml-documentation/mvdxml-schema-documentation-v1-1-draft
10. LegalRuleML_TC: LegalRuleML. OASIS (2012), https://lists.oasis-open.org/archives/legalruleml/201208/msg00040/LegalRuleML-palmirani2012_-RuleML2012v3.pdf
11. Dimyadi, J., Solihin, W., Eastman, C., Amor, R.: Integrating the BIM rule language into compliant design audit processes. In CBI W78 2016 (33rd CIB International Conference on IT in Construction) (2016)
12. Daum, S., Borrmann, A.: Checking spatio-semantic consistency of building information models by means of a query language. In Proceedings of the International Conference on Construction Applications of Virtual Reality (2013)
13. Lee, J.K.: Building environment rule and analysis (BERA) language and its application for evaluating building circulation and spatial program. Georgia Institute of Technology (2011)
14. Park, S., Lee, Y.-C., Lee, J.-K.: Definition of a domain-specific language for Korean building act sentences as an explicit computable form. J. Inf. Technol. Constr. **21**(26), 422–433 (2016)
15. Solihin, W.: A simplified BIM data representation using a relational database schema for an efficient rule checking system and its associated rule checking language. Georgia Institute of Technology (2015)
16. Preidel, C., Daum, S., Borrmann, A.: Data retrieval from building information models based on visual programming. Vis. Eng. **5**, 18 (2017)
17. Zhang, J., El-Gohary, N.M.: Integrating semantic NLP and logic reasoning into a unified system for fully-automated code checking. Autom. Constr. (2016)
18. Hjelseth, E., Nisbet, N.: Capturing normative constraints by use of the semantic mark-up (RASE) methodology. In CIB W78 2011 28th International Conference-Applications of IT in the AEC Industry (2011)