Standardized Computable Rules

Developed in cooperation with the National Office of Building Technology and Administration, and Statsbygg

Standards Norway, December 2009
Standardized Computable Rules

Authors: Håvard Bell (Catenda AS), Lars Bjørkhaug (Catenda AS) and Eilif Hjelseth (Standards Norway)

Published by:
Standards Norway
Strandveien 18
P.O.Box 242
NO-1326 Lysaker
Norway

Telephone + 47 67 83 86 00
Telefax + 47 67 83 86 01

info@standard.no
www.standard.no

ISBN 978-82-7202-673-7
Standardized Computable Rules

Authors:
Håvard Bell, Lars Bjørkhaug and Eilif Hjelseth

Project Partners:
National Office of Building Technology and Administration

Standards Norway

Statsbygg

Oslo, Norway 2009-12-04
Version 0.9
Table of Contents

1 Introduction ........................................................................................................................................... 1
   1.1 Background ........................................................................................................................................ 1
      1.1.1 Initiators and Stakeholders ........................................................................................................... 1
      1.1.2 Project Objectives ........................................................................................................................ 1
      1.1.3 Terminology ................................................................................................................................... 2
         Definition of Terms ............................................................................................................................ 2
   1.2 Description of Challenge .................................................................................................................... 3
      1.2.1 Stakeholders Interest ..................................................................................................................... 5
      1.2.2 Example of Rule Sources ............................................................................................................. 5
   2 State of the Art ....................................................................................................................................... 6
      2.1 Existing Process in Standardization Work ......................................................................................... 6
         2.1.1 ISO’s Project Stages ...................................................................................................................... 6
         2.1.2 Development Tracks .................................................................................................................. 8
      2.2 Existing Computable Rule Formats ................................................................................................ 9
   2.3 Software ............................................................................................................................................... 10
      2.3.1 SMARTcodes ................................................................................................................................ 11
         Commercial Background ..................................................................................................................... 11
         Rule Developing Process ..................................................................................................................... 12
         Technology .......................................................................................................................................... 12
         Conclusions ....................................................................................................................................... 12
      2.3.2 SOLIBRI Model Checker ............................................................................................................. 13
         Commercial Background ..................................................................................................................... 13
         Rule Developing Process ..................................................................................................................... 14
         Technology .......................................................................................................................................... 14
         Conclusions ....................................................................................................................................... 14
      2.3.3 EDM Model Server ...................................................................................................................... 15
         Commercial Background ..................................................................................................................... 15
         Rule Developing Process ..................................................................................................................... 15
         Technology .......................................................................................................................................... 15
         Conclusions ....................................................................................................................................... 16
      2.3.4 House Designer ........................................................................................................................... 17
         Commercial Background ..................................................................................................................... 17
         Rule Development Process .................................................................................................................. 17
         Technology .......................................................................................................................................... 17
         Conclusions ....................................................................................................................................... 18
      2.3.5 ePlan Check with Fornax .............................................................................................................. 19
         Commercial Background ..................................................................................................................... 19
         Rule Developing Process ..................................................................................................................... 19
         Technology .......................................................................................................................................... 19
         Conclusions ....................................................................................................................................... 20
      2.4 Computable Rule Frameworks and Concepts ................................................................................... 20
      2.5 Experience from Vendors ............................................................................................................... 21
   3 The Rule Definition Process .................................................................................................................. 23
      3.1 Importance of Process ....................................................................................................................... 23
      3.2 Iterative and Transparent .................................................................................................................. 23
      3.3 Cross Disciplinary Collaboration .................................................................................................... 24
      3.4 Importance of Human Interaction ................................................................................................... 24

2
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5 Recommended Process Steps</td>
<td>25</td>
</tr>
<tr>
<td>3.5.1 Scope the Rule</td>
<td>27</td>
</tr>
<tr>
<td>Deliverables</td>
<td>27</td>
</tr>
<tr>
<td>3.5.2 Structure the Rule Source</td>
<td>28</td>
</tr>
<tr>
<td>Deliverables</td>
<td>28</td>
</tr>
<tr>
<td>3.5.3 Creation of Computable Rules</td>
<td>29</td>
</tr>
<tr>
<td>Deliverables</td>
<td>29</td>
</tr>
<tr>
<td>3.5.4 Verification</td>
<td>30</td>
</tr>
<tr>
<td>Deliverables</td>
<td>30</td>
</tr>
<tr>
<td>3.5.5 Implementation</td>
<td>31</td>
</tr>
<tr>
<td>Deliverables</td>
<td>31</td>
</tr>
<tr>
<td>3.5.6 Testing</td>
<td>32</td>
</tr>
<tr>
<td>Deliverables</td>
<td>32</td>
</tr>
<tr>
<td>3.5.7 Certification</td>
<td>32</td>
</tr>
<tr>
<td>Deliverables</td>
<td>32</td>
</tr>
<tr>
<td>3.6 Weighting of Rules</td>
<td>33</td>
</tr>
<tr>
<td>3.7 Piloting NS 3940:2007</td>
<td>33</td>
</tr>
<tr>
<td>3.7.1 Define the Rule Scope</td>
<td>33</td>
</tr>
<tr>
<td>3.7.2 Restructure the Rule Source</td>
<td>34</td>
</tr>
<tr>
<td>3.8 Summary</td>
<td>36</td>
</tr>
<tr>
<td>4 Library of Rules</td>
<td>37</td>
</tr>
<tr>
<td>4.1 Cross Referencing</td>
<td>37</td>
</tr>
<tr>
<td>4.2 Reuse of Existing Rules</td>
<td>38</td>
</tr>
<tr>
<td>5 Technology</td>
<td>39</td>
</tr>
<tr>
<td>6 Data Quality</td>
<td>41</td>
</tr>
<tr>
<td>7 Business Models</td>
<td>42</td>
</tr>
<tr>
<td>7.1 Availability of Skill</td>
<td>43</td>
</tr>
<tr>
<td>7.2 Cost of Rule Development</td>
<td>43</td>
</tr>
<tr>
<td>7.3 Software Support</td>
<td>44</td>
</tr>
<tr>
<td>7.4 Sustainability</td>
<td>44</td>
</tr>
<tr>
<td>7.5 Responsibilities and Risk</td>
<td>44</td>
</tr>
<tr>
<td>7.6 Possible Business Models</td>
<td>44</td>
</tr>
<tr>
<td>7.6.1 Software Vendors Pay for Certification</td>
<td>45</td>
</tr>
<tr>
<td>7.6.2 Rule Owners Pay</td>
<td>45</td>
</tr>
<tr>
<td>7.6.3 Subscription Based</td>
<td>45</td>
</tr>
<tr>
<td>7.6.4 End Users Pay for Software</td>
<td>45</td>
</tr>
<tr>
<td>7.6.5 End Users Pay for End Result</td>
<td>45</td>
</tr>
<tr>
<td>7.6.6 Public Authorities Pays</td>
<td>45</td>
</tr>
<tr>
<td>8 Further Development</td>
<td>46</td>
</tr>
<tr>
<td>8.1 Overall Project Recommendations</td>
<td>46</td>
</tr>
<tr>
<td>8.1.1 Process Recommendations</td>
<td>46</td>
</tr>
<tr>
<td>8.1.2 Technology Recommendations</td>
<td>47</td>
</tr>
<tr>
<td>8.1.3 Business Recommendations</td>
<td>47</td>
</tr>
<tr>
<td>9 Conclusion</td>
<td>48</td>
</tr>
<tr>
<td>References</td>
<td>49</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>51</td>
</tr>
<tr>
<td>1 Appendix/Appendix</td>
<td>52</td>
</tr>
<tr>
<td>1.1 Methodology</td>
<td>52</td>
</tr>
<tr>
<td>1.2 Theory and Methods Adaptable for Use in the AEC Industry</td>
<td>53</td>
</tr>
<tr>
<td>1.2.1 Theoretical Foundation for Development of Rules</td>
<td>53</td>
</tr>
<tr>
<td>1.2.2 Classification in the AEC Industry</td>
<td>54</td>
</tr>
<tr>
<td>1.2.3 Concepts for Mapping Ontology - IFD</td>
<td>54</td>
</tr>
</tbody>
</table>
1.2.4 Theories and Methods for Finite Domains .............................................. 55
1.3 Methods for the AEC Industry ................................................................. 56
  1.3.1 From English to Rulish Versions of Standards ................................. 56
  1.3.2 ISO Supported System for Domain Knowledge ................................. 57
  1.3.3 MOKA and Participants ................................................................. 58
  1.3.4 Using the IFC Constraint Model ....................................................... 58
  1.3.5 Translation Versus Transformation ................................................. 59
  1.3.6 Official Computable Rule-sets - Rulish Version of Standards ............. 59
  1.3.7 Example of Projects Using a Formal Methodology ............................. 60
References - Annex/Appendix ................................................................. 60
2 Experience From Software Vendors .......................................................... 63
  2.1 Experience from AEC3 ....................................................................... 63
    2.1.1 Process Experience ................................................................. 65
  2.2 Experience from Jotne EPM Technology ............................................. 65
  2.3 Experience from Solibri ................................................................. 66
  2.4 Experience from SINTEF Building and Infrastructure ......................... 67
1 Introduction

1.1 Background

1.1.1 Initiators and Stakeholders
This investigative project was initiated by BE, Statsbygg and Standards Norway. Catenda AS was hired to join the working group, act as editor for this report and contribute with expertise in the field.

1.1.2 Project Objectives
This investigative project goal is to help the main project to determine its feasibility and potential size, and clearly describe the state of the art for the main project can build further upon.

The stakeholders see a real benefit to checking building information models (BIM) for compliance to their own set of rules and guidelines. Rule owners should be able to create and maintain such rules, independent of any proprietary tool, software vendor or business. Software vendors should be able to implement standardized computable rules and then be certified. Three areas must be clarified for this to take place, and thus the goals for the main project should be:

• Standardize the computable rule development process
• Decide on a formal standardized computable format for delivery of rules
• Explore and clarify the business processes related to creating, implementing and maintaining computable rules

In this preliminary report we experimented with a standardized rule example, based on NS 3940:2007 Areas and volumes of buildings from Standards Norway, a real standard from Standards Norway. This was done to gain experience that would further quality assure our conclusions in this report.
1.1.3 Terminology

Definition of Terms

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>buildingSMART</td>
<td>A collection of ISO standards related to information flow in the construction industry. As of this writing the standards are: IFD, IFC and IDM</td>
</tr>
<tr>
<td>IFD</td>
<td>International Framework for Dictionaries (ISO 12006-3), a framework for ontologies, terminology, classifications, properties and their relationships</td>
</tr>
<tr>
<td>IFC</td>
<td>Industry Foundation Classes (ISO 16739), an exchange format for building information</td>
</tr>
<tr>
<td>IDM</td>
<td>Information Delivery Manual (ISO 29481-1), a framework to describe what information should be exchanged when</td>
</tr>
<tr>
<td>BIM</td>
<td>Building Information Model(ing)</td>
</tr>
<tr>
<td>rule</td>
<td>A rule definition is a true or false (Boolean) expression that contains various conditions to evaluate data records. Rule definitions can contain simple tests or complex nested Boolean conditions. (IBM 2009)</td>
</tr>
<tr>
<td>rule checking</td>
<td>The process of verifying that a rule is satisfied or not, based on a given set of information to which the rule applies</td>
</tr>
</tbody>
</table>

Table 1: List of Terms

Type of Sources from which Rules are Made

There are several types of rules on which it is possible to build standardized computable rules. For clarity we have narrowed this down to four types of various sources of rules:

- Laws/Regulations/Rules (e.g building codes, laws or local regulations)
- Standards
  - National level (for Norway – NS)
  - European level – EN
  - Global level – ISO
  - or European level adopted as national standard, NS-EN, Global level adopted as national standard – NS-ISO
- Business methods (company defined policies, methods, regulation, “way of doing business”; selecting and giving priority to different solutions)
• Guides (e.g. guides to laws and building codes, internal company guidelines or guidelines based on research, experience and knowledge)
• Demands/Requirements (e.g. demands from end user, requirements from building codes or requirements from knowledge systems)

These can also be classified by the strength of the rules. Some are pass-or-fail, some give advice with varying degrees of emphasis. Some, such as the clients brief, are aspirational, and may be refined during the design process.

1.2 Description of Challenge

Each of the Stakeholders have one or more set of rule sources as described above. For example, Standards Norway maintain a large set of Norwegian Standards (NS) and act as facilitators for the ISO standards development. These standards are paper based, and in that sense passive knowledge. The user need to read, understand and then possibly apply these standards to her work. With the introduction of open standards based BIM, it is possible to make the knowledge inherent in any standard, more active. A computer can apply the rules embedded in a standard on a computer based model of the real world, e.g. a building, and report the results with regards to compliance, back to the user.

40% of building defects in Norway (and similar number for other European countries) can be related to mistakes or omissions in the design phase (Ingvaldsen, T. 1994, 2001). It is a general belief that efficient application of rule checking can reduce this number drastically.

Shifting the traditional design process towards cheaper changes earlier in the design phase require new tools, standards and forms of collaboration. Rule checking is seen as one way to facilitate this change.
In this report we have looked more closely at the Norwegian standard for area calculation (NS 3940). This is a document describing in detail how various types of areas for a building should be measured. For example, when selling an apartment the bigger area the better, but are you allowed to claim area under low roofing as part of the total area for the apartment? The standard is recently completed, and automatic rule checking was part of the considerations the authors considered when writing the rule. At the same time, the geometry is complicated enough to challenge the project and hopefully raise generic questions that are important to answer.

Obviously, such a narrow sample from a standard, are not able to catch the whole breadth of the challenges associated with the task at hand, but we believed it was important to keep a small scope in order to reach some kind of conclusion to build upon in the main phase of this project. One experience is the high demand for precision of defining relevant information. One example is that “BTA”, gross area, is calculated based the outside of the outer paneling or plaster of the wall. Length measurements relates to the “outside” of the load bearing core of the wall. The difference in most building are about 1 - 5 cm. The first information is much harder to determine from a model – need to use a
calculation algorithm on for each wall. Calculation of area based on the existing functionality is relatively easy for use. This accuracy is only (most) needed for building applications, and here can overrun of limited area result in rejection of application.

### 1.2.1 Stakeholders Interest

It is paramount to the stakeholders that their future library of computable rules are independent of software vendors, proprietary platforms and uncommon skill set. Just like anyone can put up a standards based web page, using a variety of different tools, with varying complexity, using readily available skill sets, expose it to the world on any number of standards based platform, it should be reasonably easy to build and maintain standardized computable rules using a wide range of standards based tools and platforms.

And just as producing good up-to-date web content is a complex and iterative process, involving different kinds of expertise, producing good, maintainable and up-to-date standardized computable rules will require a good and transparent process.

Bringing knowledge from an experts brain and into a computable rule is today a long, labor intensive, complex and costly process. This must be made quicker, cheaper, more transparent and less complex. To achieve this a combination of standards, standard based tools and platforms, standards based processes and skill set of achievable complexity.

### 1.2.2 Example of Rule Sources

In this report we will refer to one particular rule source, which have been chosen by the stakeholders to be suitably representative, without being too complex. This is The Norwegian Area and Volume Calculation Standard NS 3940:2007 Areas and Volumes of Buildings (NS 3940 Areal- og volumberegninger av bygninger). It was released as version 3 late 2007, and will be referred to as NS 3940:2007 throughout this report.

In addition to this rule source, we invent a very simple but possibly practical rule, to be used throughout this project to further highlight challenges and possibilities with standardized computable rules. We call it “HasProject”, and the rule is: *a BIM must have a project defined.*
2 State of the Art

Current state of the art for computable rules are most visible in software. In terms of the main projects objectives we will describe the state of the art for the following areas:

- Standardized development processes
- Standardized computable format for computable rules
- Current business models for computable rules (mainly software)

Standardized development processes are numerous and varied. However, the standardized development processes for national and international standards are well documented and well tried. This is thus a suitable baseline for a development process for standardized computable rules.

Use of BIM enables the rules to be executed on the information defined in the model. This is time and cost reducing factor compared to manual input of information in the rule checker software.

There are many ways of giving a computer a set of well documented instructions on a standard format. The challenge here is to choose the most suitable format, and at which level the format must apply.

Finally, current business models that involve computable rules, are mostly focused around existing software, and as far as this investigation is concerned not around larger “eco-systems” in which standardized computable rules play an important part.

2.1 Existing Process in Standardization Work

2.1.1 ISO's Project Stages

ISO’s deliverables are developed through a sequence of project stages. Each stage has its name, but very often the stages are identified by using the acronyms that are associated with each stage. For example, the enquiry stage, during which a Draft International Standard is produced, may be identified by using the acronym for this document, namely DIS. The complete list listed in Table 2: ISO Development Stages.
<table>
<thead>
<tr>
<th>Stage Name</th>
<th>Product Name (Document)</th>
<th>Acronym</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preliminary Stage</td>
<td>Preliminary Work Item (Project)</td>
<td>PWI</td>
</tr>
<tr>
<td>Proposal Stage</td>
<td>New Proposal for a Work Item</td>
<td>NP</td>
</tr>
<tr>
<td>Preparatory Stage</td>
<td>Working Draft(s)</td>
<td>WD</td>
</tr>
<tr>
<td>Committee Stage</td>
<td>Committee Draft(s)</td>
<td>CD</td>
</tr>
<tr>
<td>Enquiry Stage</td>
<td>Draft International Standard</td>
<td>DIS</td>
</tr>
<tr>
<td>Approval Stage</td>
<td>Final Draft International Standard</td>
<td>FDIS</td>
</tr>
<tr>
<td>Publication Stage</td>
<td>International Standard</td>
<td>IS</td>
</tr>
</tbody>
</table>

*Table 2: ISO Development Stages*
2.1.2 Development Tracks

Time limits have been introduced to reduce the risk of investing resources in projects that have insufficient hope of success. It is recognized that some projects need more time than others, so three tracks are specified: accelerated, default and extended.

Figure 2: The Six ISO Development Stages
2.2 Existing Computable Rule Formats

Any programming language can be considered a standardized computable rule format. It is a standardized way of giving a computer a set of instructions to perform. Usually, a programming language, for example C++, is developed in an open standardized process, much like the one mentioned in 2.1 Existing Process in Standardization Work, to produce a standardized programming language. All instructions and their expected output are well defined. A compiler is then used to translate the set of instructions into executable code, or programs, on a given computer platform.

The closer a programming language is to the underlaying computer platform on which it is supposed to be used, the more verbal it generally is. For example, see Table 3, to see the different set of standardized instructions needed to produce the same output, “Hello World” (Wolfram 2009).

buildingSMART is the open international standard available for the AEC Industry when exchanging information. The core of buildingSMART is the IFC exchange format, a large data model allowing anyone to describe precisely a building model. The underlaying standard to describe this exchange format is the EXPRESS language. The IFC standard also includes a set of entities to describe constraints to be put on the building model itself, The IFC Constraint model. This is a powerful and standardized way of describing computable rules to be applied to a building information model based on the IFC standard.

The challenge for the main project is to investigate if the existing solutions cover the needs for standardized computable rules format to be developed in the main project.
Table 3: Same output, but different set of instructions in different programming languages (Wolfram 2009)

Very often, the compiler is a commercial product, developed for profit and sold to users who need to build programs.

2.3 Software

Various software exist today that does automatic rule checking applied to a standardized BIM. This can be compared to the main business models in the early days of software development, where people made money in two areas:

- Compilers that compiled standardized instructions from a programming language into executable programs for a given computer platform
- Executable programs that performed a certain task (or a set of tasks closely related)

The first business model was mainly targeted at developers. In our case this
would be analogous to the rule developers. The second model is the one most visible today for computable rules in the AEC Industry. Users buying rule checking software. A great review of current rule checking software is done in Eastman 2009, please consult this for additional details to the list below.

2.3.1 SMARTcodes

SMARTcodes is a joint project with the International Code Council (ICC), AEC3 and Digital Alchemy. Some of the tools developed has been used within ICC, but is not available commercially at this point. This is probably due to the fact that the important part of the SMARTcodes is the process itself, not so much the tools involved.

Commercial Background

SMARTcodes is a joint project with the International Code Council (ICC), AEC3 and Digital Alchemy. Some of the tools developed has been used within ICC, but is not available commercially at this point. This is probably due to the fact that the important part of the SMARTcodes is the process itself, not so much the tools involved.
Rule Developing Process

The SMARTcodes project is more about a process than tools, where the process will capture the essence of a building code and convert this into a computable rule. These are the steps behind the SMARTcodes project:

- Start with original building codes, in textual format
- Markup the text in the building codes with colors, highlighting the subject, the verb (action), context, and so forth, according to the SMARTcode protocol
- The markup text is made into structured XML, keeping the original text
- The structured XML is then converted into computable rules expressed in the IFC Constraint model
- The IFC Constraint model, is in effect a standardized computable rule format, usable for any rule engine capable of reading an IFC file
- A one-to-one mapping between the final standardized computable rule format, the IFC Constraint model, and the original building code text, the structured XML document, is kept

Technology

There are two important technologies in the SMARTcodes project:

- The SMARTcode protocol
- Algorithms converting color coded markup to IFC Constraint rules

The SMARTcode protocol defines which textual part should have which color, the algorithms convert this into computable rules that can be executed by any software understanding the IFC model.

Conclusions

SMARTcodes represents the most process oriented approach to rule checking among the rule checking offers available today. The greatest challenge with this approach is its applicability to various kinds of text from different rule sources. Although the approach has been tested\(^1\) on GSA courts design guides and Korean

\(^1\) Example of code for which SMARTcodes has been successful: “502.5 Moisture control. (Mandatory). All framed walls, floors and ceilings not ventilated to allow moisture to escape shall be provided with an approved vapor retarder having a permeance rating of 1 perm \((5.7 \times 10^{-11} \text{ kg/Pa} \cdot \text{s} \cdot \text{m}^2)\) or less, when tested in accordance with the desiccant method using Procedure A of ASTM E 96. The vapor retarder shall be installed on the warm-in-winter side of the insulation. Exceptions: Buildings located in Climate Zones 1 through 3 as indicated in Figure 301.1 and Table 301.1. In construction where moisture or its freezing will not damage the materials. Where other approved means to avoid condensation in unventilated framed wall, floor, roof and ceiling cavities
codes in addition to ICC’s codes, it is not tested on Norwegian functional based rules, knowledge bases or standards. The following key points are important for this project:

- Direct link between rule source and computable rule
- Allow multi disciplinary work through adapted tools and methodology
- Rule checking platform agnostic
- Direct link with rule source saves labour

2.3.2 SOLIBRI Model Checker

![Figure 5: Example of Solibri Model Checker Desktop Window](image)

Commercial Background

SOLIBRI Model Checker is offered in the marked by SOLIBRI, a Finish software company. Their business model is to offer software that can compare two IFC based BIMs and check for collisions or other conflicting configurations. They also offer a set of built-in rules, and a rule configuration based on parameters. For example, the user can choose the size of an accessibility circle necessary to fit in a restroom.

are provided.”
Rule Developing Process

If the user is interested in new rules, rules that cannot be built with the built-in rule configuration tool, it must be custom made. SOLIBRI will then work with the user to ensure the correct interpretation of the rules, and build it into its software. This requires experts from SOLIBRI as well as experts on the rules themselves. In the process, SOLIBRI will try to parametrize the rules as much as possible.

Technology

SOLIBRI is based on Java, and they have built their own rule engine around this. New rules must be coded up, using their library and regular Java code. They provide the user with a clever view of the BIM and the rule conflicting parts, using advanced 3D modelling.

Conclusions

SOLIBRI Model Checker is traditional in the sense that most of the rules are hard coded into the software, and there is no configurable user friendly language in which to specify new rules. Any rules written for SOLIBRI Model Checker cannot readily be adopted in other software as well.
2.3.3 EDM Model Server

Commercial Background

EDM Model Server is sold in the marked by Jotne EPM Technology. It is a powerful model manipulating tool, pitching itself as a complete solution for model repositories, including ensuring data integrity and quality. The model server consists of an object-based database, and a standardized set of tools to interact with the data in the database.

The tools are complex to use, and require knowledge of EXPRESS and EXPRESS-X, a standardized object-oriented query language, the same used to describe the buildingSMART standards, but gives the user complete flexibility. Computable rules can be written in EXPRESS-X and compiled and executed directly on EDM Model Server. Rules written in this form, can be used in any other software that understands EXPRESS-X.

Rule Developing Process

Like SOLIBRI Model Checker, the EDM Model Server comes with a set of built-in rules. To develop new rules, Jotne EPM Technology will aid the user in creating the necessary rules, again using experts on EXPRESS from Jotne EPM Technology and experts on the original rule sources.

Technology

EDM Model Server is based on the open international standard EXPRESS, so
your data can be imported and exported using open standards. Computable rules expressed in the same standardized language as the IFC model can be executed directly on the EDM Model Server. This provides a powerful and flexible platform, but also a set of tools that require a high level of expertise.

**Conclusions**

EDM Model Server is a very powerful platform to build rules on. It allows the user to build computable rules on an open standards format and provides an environment in which these can be executed. The platform is flexible, but also complex, and requires highly skilled professionals to run.
2.3.4 House Designer

![Figure 7: Example of House Designer Desktop Window](image)

**Commercial Background**

House Designer is one of the offerings from Selvaag BlueThink\(^2\). The core of the software is a complex rule engine, written in Lisp, a programming language well suited for interdependent rules. House Designer is used within Selvaag Group and sold commercially on the marked. The software comes with a set of built-in rules.

**Rule Development Process**

In order to develop new rules for House Designer, House Designer programmers must work together with the domain experts to develop new built-in rules. Any new rule is then customized for House Designer and not applicable in any other software.

**Technology**

House Designer is built around a Lisp engine. The software also provides a specialized viewer, to combine information about broken rules with a visual image of the building. Most of the code around the Lisp engine is written in Java.

---

\(^2\) Selvaag BlueThink is currently for sale in the marked, so its future is unknown at the time of this writing.
Conclusions

Unlike the other software listed here, House Designer is able to generate buildings from rules, not just passively checking if an existing building satisfy certain rules. This technology relies heavily on pre configured objects and well defined building blocks. Rules adapted for the House Designer platform cannot be adapted in other software.
2.3.5 ePlan Check with Fornax

![Example of ePlan Check with Fornax Browser Window](image)

**Commercial Background**
ePlan Check is a code checking platform run by the Singapore Authorities. The platform uses EDM Model Server, and their rule engine is built using Fornax, a specialized library built in C++, allowing a computer programmer to build computable rules. Fornax was on offer to interested parties, but its status is uncertain at the moment.

**Rule Developing Process**
In order to build new computable rules using Fornax, a computer expert in C++ must sit down with a domain expert and build the rules. This computable rule will then work with the Fornax library, but not on any other platform. It is unclear why the Singapore Authorities elected to use Fornax in addition to the EXPRESS based solution on offer with the EDM Model Server.

**Technology**
ePlan Check with Fornax is based on the EDM Model Server (see 2.3.3) in addition to a C++ based library customized on top of EDM Model server, allowing computable rules to be built on top of the platform.
Conclusions
This is an early, and working, rule based engine, where the purpose of the platform was to allow Singapore builders to check their models against the Singapore building codes, automatically and whenever they chose. Rules developed for this platform cannot be transferred.

2.4 Computable Rule Frameworks and Concepts
There seem to be no widely used formal methodology to generate computable rules in the construction industry. The most common form is to have computer programming experts examine the rule source, sometimes with access to the domain expert behind the rule source, and generate computer code. This methodology can be “practical” for implementing small number of simple rules and testing possibilities with BIM / IFC based technology. But it is no way near the needs of the stakeholders in this case.

The exception to this is the SMARTcode project, which has tried to enable the domain experts to form the computable rules themselves, through simple tools. This has proven successful for building codes that was made up from reasonably structured text. The SMARTcode project do not include methodology to cross reference rules from various libraries, since this is already taken care of by the ICC rule writing methodology itself.

Figure 9: 3-tier knowledge level (Hjelseth 2009) illustrates a framework for development rule checking systems, that can be used to guide further development in this field.

The intention with Figure 9 is to illustrate that it is possible to use a tier approach, the most important is to clarify and standardize the interface between the tiers (layers)- not what is "inside" each tier.
Eastman 2009 has used a framework to compare different rule checking software. The framework can also be seen as a methodology for rule checking itself, as illustrated in Figure 10.

**Figure 10: Framework for Rule Checking Steps by Eastman 2009**

### 2.5 Experience from Vendors

Part of the surveys conducted in this report included a few online meetings with all the software vendors we so far have recognized with relative long experience with rule checking on buildingSMART based BIMs. The software vendors included in these sessions are the following:

- AEC3
- CRC
- Jotne EPM Technology
- Selvaag BlueThink
- Solibri

Possibly, we should also have included Autodesk Navisworks, but were unable to do so at this stage.

Three of the software vendors were also kind enough to include some written thoughts on their experience and opinions in the area going forward, and this
Furthermore, they were all very positive to an international collaboration effort in this field going forward.

In addition to the more elaborate points given in the Appendix/Annex, a few issues were raised in the online meetings with the vendors:

- Important to know which rules have been applied and why
- Data quality is essential
- Importance of separation of data quality checking and rule checking
- Importance of separation of rule making and rule checking
- The need for iterative processes

Please find a more detailed description from the vendors themselves in Appendix/Annex 2 Experience From Software Vendors.
3 The Rule Definition Process

This pilot project has convinced us that the rule definition process is crucial to creating standardized computable rules. There already exist well known and efficient processes to define rule sources, such as the clearly defined processes to make national and international standards. The process to generate standardized computable rules should look at these well known standards and reuse their experience as much as possible. For example, can we reuse something from the IDM process description? Can we use whole or part of the ISO standardization process?

Perhaps the whole project should focus more on the process, as a tool to generate more automatic checks during building information modeling and the life cycle of a building.

3.1 Importance of Process

Our discussion and research so far has given us strong indication that figuring out the process for neutral computable rule creation, is the most important feature of this project.

The various disciplines, perspectives and the complexity often implicitly inherit in rule sources lead us to believe that this is not only a technology issue, but perhaps most of all a process issue.

The complex path the various pieces of knowledge travels before they turn into computable rules, and the importance of accountability for every computable rule that will be applied, dictates a clear, well defined, yet flexible and transparent process.

3.2 Iterative and Transparent

Due to the complexity foreseen in the rule making process, and learning from other similar processes, it seems natural to apply an agile style of development. An iterative process, with frequent milestones can avoid derailing and constantly check the feasibility of the development. Software development, essentially a large rule making process, has had great success with this kind of methodology.

This iterative approach is also essential in the application of the computable rules. A building process is in itself highly iterative, with frequent updates, feedback loops and a constant refinement of the design. Constantly increasing the level of details and complexity. Thus the computable rules should be applied frequently along the constant refinement of the design, adding helpful information to the feedback loops.
In a similar fashion, the computable rules development process should aim at
iterations that constantly refine and details the computable rules themselves. For
example in terms of the Area Calculation Standard, it would be reasonable to
start with the simplest case with well defined and well behaved square box of a
building. The next step could be to add rules that also considers tilted roof tops.
Then consider cases with rooms that are not considered living quarters, and so
forth.

At the same time, all assumptions, decisions and background information must
be clearly documented. There should be a trail of unambiguous information
behind every computable rule, preferably ending up in a small, but well scoped,
clearly defined and well documented piece of knowledge.

In many cases, it is also important to document what the computable rule do not
cover, and in which cases it will not apply.

The end user need to trust that the rules he applied are relevant and correct. If
he can replay all the steps used to reach this particular rule he will have a great
source of documentation added to his quality assurance procedures. Assuring
transparent and well documented assumptions and scope, will ensure that the
trust threshold is lower.

### 3.3 Cross Disciplinary Collaboration

The rule making process must include experts from various fields. For example,
the authors of a standard should be present to ensure that the intention of the
standard is captured, and not necessarily how the coincidental computable rule
makers understood the text at the time.

Furthermore, it cannot be expected from an expert on standardized Area
Calculation in the building domain, to be well versed in the intricate details of
computer implementation and computer programming.

It would thus be advantageous to develop tools and procedures allowing each
expert to work within his field of expertise, but in such a way that it will benefit
the other experts taking part in the process as well. For example, often will the
questions raised by a computer programming expert, help the author of the
standard to further clarify his intention. A process both experts will gain from.
Applying tools and the right amount of formalities into this could be of great
value.

### 3.4 Importance of Human Interaction

Many rules will need some kind of human interaction. Either to verify results,
double check, or make decisions the computer are not able to do.

Human interaction should be focused once an issue has been discovered. Reports
should include clear reference to, or quotation of, the source clause and also to
the object in failure both graphically and by description. Given a proper logical
base, any result can be described in terms of its immediate cause and the
decisions leading up to it.

This reporting can optionally be supplemented by systematic listing of
alternative resolutions: for example a thermal failure could be resolved by re-
specifying the object performance, or triggering an exception, or by reclassifying
the component as not part of the envelope, or even re-designating the building
type.

Just like Clash detection, Compliance checking discovers a conflict between two
objects: in the one case the conflict is between two building objects, in the other
it is between a regulatory objective and a building object. Even simple model
review and markup documents a conflict between the objects in the model and
the reviewers expectation. Each of these design management events can be
captured and resolved.

Human interaction can also add to the trustability of the rule checking, again
lowering the threshold of trust for the user.

3.5 Recommended Process Steps

Below are the recommended process steps to generate standardized computable
rules. Each steps identify what should be done, what should be delivered from
this step, and who the main actors in this step should be.
<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
<th>Deliverable</th>
<th>Actors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Scope, define the rule source, assumptions and terminology</td>
<td>Document defining sources, scope, assumptions and terminology</td>
<td>Rule experts, appointed by rule owners</td>
</tr>
<tr>
<td>2</td>
<td>Structure the rule source according to logic and terminology</td>
<td>Document defining the rule source clearly according to document from first stage</td>
<td>Rule experts in collaboration with computable rule experts</td>
</tr>
<tr>
<td>3</td>
<td>Create the computable rules, possibly build upon existing rules</td>
<td>Computable rules on standardized format</td>
<td>Rule experts and computable rule experts</td>
</tr>
<tr>
<td>4</td>
<td>Verify the computable rules</td>
<td>Report with verification results</td>
<td>Computable rule experts with rule experts, different from makers of the computable rules</td>
</tr>
<tr>
<td>5</td>
<td>Implement the computable rules</td>
<td>Running rule checking software</td>
<td>Software developers</td>
</tr>
<tr>
<td>6</td>
<td>Test the computable rules</td>
<td>Test results</td>
<td>Rule experts</td>
</tr>
<tr>
<td>7</td>
<td>Certify implementations</td>
<td>Certified software</td>
<td>Certification authorities</td>
</tr>
</tbody>
</table>

Table 4: Recommended process steps

It is also important to note that these steps are not necessarily completed in succession, but in a more iterative fashion. For example, the scope can be coarsely defined by some stakeholder; then a committee formed to refine the scope, before defining part of all the computable rules. These computable rules are then verified, before more computable rules are defined, and so forth.

This iterative process should produce more and more refined deliverables, and ensure that the process stays on track, and meets the stakeholder's need.

It is also important to see this process in parallel with the traditional standardization process described in 2.1.1 ISO’s Project Stages. A standard, and its standardized computable rule set, can both be deliverables at the end of the existing processes. The above recommended process steps is thus performed within the well defined process stages in use today.

Furthermore, depending on the complexity and size of the set of computable rules produced, the amount of resources put into every step might vary. For example, for our own HasProject rule some of these steps might be omitted (e.g 1., 4. and 7.) or completed very quickly (e.g 2., 3., 5. and 6.). While for a more complex set of computable rules, like NS 3940:2007, it is necessary to spend considerable resource on each step to ensure the usefulness of the end result.
3.5.1 Scope the Rule

Even a relatively clear standard such as the NS 3940:2007, will benefit from a clearly defined scope in terms of derived computable rules. For example, should these rules apply to all kinds of complex buildings? Or should the first iteration and computable rule set focus on simple dwellings? And what exactly is meant by “simple dwellings”? Any other standard dependency should also be made clear.

All the assumptions, the validity area of the produced computable rules, the intention of the work, etc, needs to be clearly and formally defined. No misunderstandings, gray areas or imprecise notions must be allowed to slip through. The computer sees the world in black or white, one or zero, and cannot cope with shades of gray. And trust in computable rules can only be achieved if they are clear and unambiguous.

Human interaction can be utilized for areas that are impossible with current technology and knowledge to define clearly, thus making it absolutely clear for the computer. For example, if a rule's assumption can only be verified by a human, the assumption can be made clear for the computer, after the human has inspected the problem and stated his result.

This step will be different depending on the rule source in question. A functional based law, might require a combination of several related standards and technical guidelines to be complete enough for step 2.

Deliverables

The output from this step is a human readable document, defining scope, assumptions and references. A complete template for such a document should be developed through experience with the process steps. An iterative process is recommended, so that the template will be clear on what is necessary for the first iteration, second iteration and so forth.

3.5.2 Structure the Rule Source

In this stage, the building domain expertise meets the computer programming domain expertise. It is likely that this stage will encourage a form of committee work, to facilitate the cross disciplinary collaboration. It should be composed of interdisciplinary knowledge, preferably experts from the field of computable rules, including creation and production in the given formalized language. Also experts from the domain in question, for example for NS 3940:2007, experts in the field of area measurement on buildings in Norway should be present. The makers of the standard itself should perhaps also be present.

The committee must be given time to learn each others point of view and expertise, enough to collaborate and use each others expertise in a productive
way. Through dialogue and work on the specific task at hand, this should be a productive form of computable rule creation.

For our HasProject rule, this step is very simple. For example, the only necessary step is to recognize that “project” is equivalent with the IfcProject entity in the IFC model, and that “defined” means that the IfcProject entity exists in the BIM.

However, for NS 3940:2007, the task is more complex. For example the document defines “Construction Area” (KA) as the area occupied by building components. Then it lists examples of building components. However, in this step, what a building component is must be completely defined. Perhaps this means a list of all possible IFC entities (objects) in a BIM, that can possibly occupy such an area. Or, perhaps even better, a complete list of generic building components and a mapping to possible IFC entities using the IFD Libraray. This step might also conclude that it is necessary to ask the user for each case.

**Deliverables**

The output from this step should be unambiguous. SMARTcodes' analogy is the color coded markup mentioned in 2.3.1 SMARTcodes. As also the experience from the SMARTcodes project states, a dictionary is essential for this step to be useful. IFD Library is the natural choice as this is part of the buildingSMART family of standards. Based on the experience from the SMARTcodes project, we recommend XML as the basis for the output of this step. However, the XML schema for the document must be developed and defined. Again, an iterative development through experience with the actual process steps is recommended.

### 3.5.3 Creation of Computable Rules

The first stage in development of computable rules is to determine whether the source can be implemented or not. This is illustrated in Figure 11: Human - PC relationship (Hjelseth 2009).
Some sources, like laws, regulations and standards, express their rules in a way that are not suitable for computable implementation. For example by listing indicators to be considered as important, but without defining a value for each indicator a computer can validate against. It is therefore important to identify these rules - and if possible, support the execution of these, through check lists and similar tools aiding the user (e.g architect, engineer and designer) achieving her goal.

The previous step should now have prepared the ground for a mostly automatic process in converting the structured rule source into computable rules. The SMARTcodes project developed algorithms to successfully complete this step automatically.

**Deliverables**

Based on the positive experience from the SMARTcodes project, we recommend the output to be produced using the buildingSMART based IFC XML format, using the IFC Constraint model in the standard. However, it is unclear how well this format will perform with regards to geometrical analysis, like the NS 3940:2007 requires, so this will need to be investigated further. Also, it is
unclear how this format will capture possible human interaction needs. Again an iterative approach should provide a low risk approach to finding a suitable format for this step's deliverable.

Figure 12: IFC Constraint Model. From the IFC Specification: "The example [...] shows how a constraint may be applied to a property within a property set. For simplicity, only the mandatory attributes are shown as asserted. It shows how a property 'ThingWeight' which has a nominal value of 19.5 kg has two constraints that are logically aggregated by an AND connection. One of the constraints has a benchmark of 'GREATERTHANOREQUALTO' whilst the second has a benchmark of 'LESSTHANOREQUALTO'. This means that the constraint must lie between these two bounding values. The relating constraint is instantiated as an objective named as 'Weight Constraint' and qualified as a SPECIFICATION constraint. The two related constraints are both specified as metrics since they can have specific values."

3.5.4 Verification

The rules produced must be verified. This should possibly be done by a larger reference group outside the core committee developing the rules. For better results, verification should also be done through actual implementation of the
computable rules, and implementation tested on a wide variety of BIMs.

The degree of verification will depend on time and allocated budget (or business opportunity) for the computable rule process, and the complexity of the rule source itself. Our HasProject rule source will require much less verification than NS 3940:2007.

Without verification, the computable rules developed may end up not being implementable or possible to compute. For example, if a rule defines width and depth of a square shaped area, the total area of the shape should be easy to find. However, the actual implementation may reveal some ambiguity in the definition of the width, so in certain circumstances, it will be impossible to find the area in question.

**Deliverables**

The output of this step in the process is a human readable document verifying all computable rules produced in the previous step. An explanation must present where the rule is not possible to verify.

### 3.5.5 Implementation

In addition to verification, it is important to actually implement the rules, in real applications as part of the development process. Through the implementation process, several ambiguities, unclear points, missing definitions and vague scope definitions can be revealed.

For example, is all the information necessary to calculate a certain area available in the BIM? In NS 3940:2007, it is not clear if all the information relating to the ground and height above ground is available in all BIMs such a rule would claim to be valid for. This is not easy to verify without an actual implementation of the computable rule.

If the output from step 3., Defining the Computable Rules, is in a well known standardized format, this part of the process should be quick and simple. However, at the time of writing of this report, it seems likely that this step will involve an iterative learning phase for all parties.

It is also possible to foresee situations where software decides to implement only part of the rules, not the complete set delivered from the previous step. This must be allowed.

**Deliverables**

The output of this step should be running software, executing the computable rules on any standardized BIM, satisfying the information requirements (IDM) dictated by the computable rules.
### 3.5.6 Testing

One of the bigger challenges to buildingSMART based BIM today is that there is “more than one way to do it”. This means that the same type of information, can be stored in different ways in a buildingSMART based BIM, depending on which software produced the BIM in question. To overcome this challenge, there are two possible choices, both of which should be used in this process:

1. Define IDMs for each standardized set of rules
2. Test the implementation on a wide range of different BIMs

This process step should therefore test the implementation, with as many different BIMs as possible within the scope and validity area of the rules in question.

With “different BIMs” we mean BIMs produced from different authoring tools, or tools that write any information back into the BIM.

### Deliverables

The output of this step is human readable documents describing the results of the tests. The documents purpose is to clarify for the developers where their software fails, and where it succeeds. Templates for these documents must be developed.

### 3.5.7 Certification

After an application or set of software tools have implemented the computable rules, they should be given the chance to certify themselves for these rules. This means that the rule creators, somehow assert that the rule engine implemented produce the desired and correct results.

Without such a certification option, there is no certain way an end user can trust the results from the rule checking performed. For example, if Software A implements the computable rules produced from NS 3940:2007, how can Customer X know that the results from Software A are correct? Did they really use “width times depth” to come up with an area, or perhaps “width times depth times a secret factor” making the results more enjoyable?

The certification process should be efficient, since it is in everyones interest that as many rule engines as possible are certified. In the long run, we believe that it is in the software implementers own best interest to create correct implementations.

Giving the user an assurance that the software performs as expected is also lowering the threshold of trust for the user. If the user cannot trust the computable rules, they will be worthless. For example, just like you need to trust your mobile phone to dial John Doe, and only John Doe, every time you push “Mr. John Doe” in the address book of your phone. If you did not trust your address
book on your mobile to do this, you would dial his number manually, and remember the phone number yourself.

Certification of software that only implemented a subset of the complete rule set delivered, must be allowed. This will lowering the threshold and increase the availability of the implementations.

**Deliverables**

The output of this step is officially certified software.

### 3.6 Weighting of Rules

Some rules may be more important than others in one setting, and vice versa in another setting. For example, “Is the room painted?”, might matter less than a certain requirement on the size. You can always paint after the building is “key-ready”, but it is difficult to change the size of rooms.

### 3.7 Piloting NS 3940:2007

The Norwegian Area and Volume Calculation Standard NS 3940:2007 Areas and Volumes of Buildings (NS 3940:2007) was chosen by the stakeholders as an example rule source. Immediately a challenge on how to approach the breaking down of the standard to computable rules, arise. There are basically two choices:

1. Assume all geometrical data exist in the BIM (areas and volumes are already calculated by some software)
2. Do not assume any geometrical data exist in the BIM (all areas and volumes must be calculated)

The first choice puts the trust of area and volume calculation to other software, the last choice put a higher demand on correct geometric representation.

Most likely a real world scenario will demand a combination of the choices. Although the original intention, we have not piloted NS 3940:2007 using the recommended process steps in this report, due to resource constraints. We have done a brief and quick first iteration of the steps, highlighting what at this point looks necessary to do at each step.

#### 3.7.1 Define the Rule Scope

We assume a small subset of simple dwellings, most commonly used in Norway. No fancy office or hospital buildings. Furthermore, we assume that all possible configurations are easily recognized and covered in the original standard document.
All geometrical data is expected to already exist in the BIM, no geometrical analysis of the BIM is required to execute the rule. Information not readily available in the BIM in the form of standardized IFC, is considered out of scope.

### 3.7.2 Restructure the Rule Source

Below is a table highlighting some of the clarifications necessary to restructure NS 3940:2007. We already realize that the assumption made in step 1, where we assumed that all geometrical data already exist in the BIM, is problematic. Especially if the BIM authoring tool do not support the NS 3940:2007 definition of for example Gross Area. Ideally, a standardized computable rule based on NS 3940:2007 cannot make any assumption on the pre calculated area and volume values in a BIM, but must analyze the geometrical data in the BIM itself, and compute the correct values according to the definition in NS 3940:2007.

This immediately demonstrates the usefulness of an iterative approach. The next step would then be to return to step 1, and change the assumptions. Which in turn will put a higher demand on step 2.
<table>
<thead>
<tr>
<th>Norwegian</th>
<th>English</th>
<th>Acronym</th>
<th>Method in BIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bebygd areal</td>
<td>Building Area</td>
<td>BYA</td>
<td>Simplification: Use the footprint of the building given by the NominalArea quantity attached to the IfcBuilding entity.</td>
</tr>
<tr>
<td>Bruksareal</td>
<td>Usable Area</td>
<td>BRA</td>
<td>Assume that subtracting the sum of GrossFootprintArea quantity attached to each IfcWall (or any subtype) with the property IsExternal equal to True from the GrossArea quantity attached to IfcBuilding will yield the correct answer</td>
</tr>
<tr>
<td>Bruttoareal</td>
<td>Gross Area</td>
<td>BTA</td>
<td>Assume GrossArea quantity attached to IfcBuilding contains the correct value</td>
</tr>
<tr>
<td>Funksjonsareal</td>
<td>Functional Area</td>
<td>FUA</td>
<td>Assume this is identical to Net Area</td>
</tr>
<tr>
<td>Kommunikasjons-areal</td>
<td>Communication Area</td>
<td>KOA</td>
<td>Challenge: Need a clear definition of what is defined by “communication area”, use of IFD Library necessary</td>
</tr>
<tr>
<td>Konstruksjons-areal</td>
<td>Construction Area</td>
<td>KA</td>
<td>Challenge: What exactly defines a “construction element”? Need IFD Library</td>
</tr>
<tr>
<td>Nettoareal</td>
<td>Net Area</td>
<td>NTA</td>
<td>Assume the NetFloorArea quantity attached to IfcBuilding contain the correct value</td>
</tr>
<tr>
<td>Bruttovolum</td>
<td>Gross volume</td>
<td></td>
<td>Assume the GrossVolume quantity attached to IfcBuilding contains the correct value</td>
</tr>
<tr>
<td>Nettovulm</td>
<td>Net volume</td>
<td></td>
<td>Assume the NetVolume quantity attached to IfcBuilding contains the correct value</td>
</tr>
</tbody>
</table>

Table 5: Non-complete restructure of NS 3940:2007, as part of step 2
The column “Method in BIM” clarifies which attributes, data and entities (objects) in an IFC based BIM will be necessary to use in order to execute the rules defined in NS 3940:2007, based on assumptions in step 1.

### 3.8 Summary

We have suggested a set of process steps, to be followed in succession, but in an iterative fashion. This means that some steps may be taken several times before some of the next steps are taken. A deliverable on each step is suggested, however the level of detail of this deliverable may vary and be adjusted to the task at hand. Figure 13 illustrate this.

Figure 13: Each step with a deliverable, indicating an iterative approach
4 Library of Rules

Rules are knowledge and likely owned by someone. The Stakeholders are sitting on top of large libraries of written knowledge. It is assumed that a lot of this information can be turned into computable rules. In the same way that the current library is maintained and improved, a future library of neutral computable rules must be maintained.

It is hard to predict what rules that will be freely available and which rules that will be the property of a particular company or organization. As an analogy most programming languages comes with a library of free functions while others have to be downloaded, freely or for a fee. What to include and what to supply as add ons depends on various factors. Some rules or functions doesn't have a business value in themselves. i.e. where to find the “Communication Area” of a building floor or the room type for a room. Another example is the “light opening” of a door for testing for wheelchair access. The latter might sound as a easy task but can involve fairly complicated geometrical analyses if the roof is low and sloped or there is another wall close to the door. If such a function will be freely available is not possible to say. In many cases you will find that commercial libraries soon will face competition from free alternatives. The same thing is likely to happen for a common rule library.

We therefore assume that we will have all kinds of libraries available:

- Private libraries, specific to a company or an organization, etc.
- Open public libraries
- Standardized libraries, quality approved and maintained by a local, national or international standardization body
- Commercial libraries, maintained and quality assured by a for profit commercial entity

4.1 Cross Referencing

Current rule sources are widely cross referenced. This is important for neutral computable rules as well. For example, when two rules contradict each other, which one should be used?

In our society a law has preference over a regulation which again has preference over a commercial company's policy or recommendation. However, a company will usually want to perform better than both the regulations and laws and thus have rules they want to have preference over laws and regulations.

A system of standardized computable rules, with interlinking libraries and
interdependence, must allow the user of these computable rules to set the correct preferences.

4.2 Reuse of Existing Rules

Already today, in existing rule sources, bits and pieces of information is reused in several places. A set of smaller computable rules that can be reused, combined in new ways to create new more complex rules, makes maintenance and creation easier.

One of IFD Library's goals is to facilitate the reuse of such knowledge across a wide range of technologies and platforms. It is important to further investigate how this can be achieved within the context of IFD Library and the rest of the buildingSMART standards.

The different libraries of computable rules in combination with IFD Library can provide a powerful platform for reuse of existing computable rules.
5 Technology

The following is a listing of the existing technology used by the software vendors we were able to talk to throughout this pilot project:

- **SMARTcodes:**
  - Code markup tool
  - XSLT transformations and IFC-ifcXML interoperability
  - Rule engines:
    - EPM and Octaga, and web server environment (AEC3 XABIO)
    - Windows command line executables and DLL’s (AEC3 Compliance1)
    - Extensions to Proprietary rule environments (Solibri + DA)
    - Extensions to standards based rule environments (ePlanCheck)

- **Platforms and Frameworks**
  - EDM (EPM Data Manager), EDM Model Server
  - EMF (Eclipse Modeling Framework)
  - Modelica
    - An open language to model logical and physical systems. In use today, but new and fresh. A set of tools, a compiler, etc.
  - Moka
    - A methodology to develop rules.

- **Program Languages (Often rules end up being implemented in some programming language)**
  - Java
    - Used in various implementations today. Object oriented.
  - Lisp
    - Common language for rule implementation.
  - C++
    - Object oriented.
  - C#
    - Object oriented. Java like
- C
  - Procedural. Often used to achieve speed.
- Fortran
  - Procedural. Often used in academia.
- EXPRESS-X
  - Used in EDM. Object oriented. STEP standard
- Prolog
- JES
- Modeling Languages
  - UML
  - SOAML
  - SysML
  - Modelica
- Query Languages
  - SPARQL
  - SQL
- Standards (Standards often have some rule computing capability)
  - buildingSMART
    - IfcConstraint model
  - Semantic Web (Recently developed many standards that ultimately aid in rule checking. Most of the standards developed by W3C)
    - OWL
    - RDF
    - RIF
      - Rule Interchange Format
6 Data Quality

In order to be able to apply neutral computable rules on a standardized BIM, the data quality of the BIM itself must be good enough for the rules to be applied. For example, in order to check the area of a wall, the model must contain enough information to deduce the height of the wall and the width of the wall. This may be found as a pre-measured quantity, or by doing a geometric analysis. Actually multiplying the height and width may be the least attractive method.

The information must also be placed in the model in its standardized place. For example in the IFC model, the Pset_WallCommon property set contains the IsExternal property. If the wall is external, this is where that information should be placed. Not in a user defined attribute somewhere else, or in another property set.

However methods can be tolerant when reading: for example if space boundaries are present, then an external element will have space boundary relationships on one side only, assuming space boundary relationships are used consistently throughout the model.

Whilst data quality is of course important, three-valued logic only requires that the unknowns be known (attr. D. Rumsfeld, abridged).

Using an IDM for a well defined rule checking process will ensure a certain level of data quality. The purpose of an IDM is to establish what kind of data, and where that data should be placed in the IFC data model, for a given process or exchange of information. For example, and IDM can demand that the area of all walls are pre calculated, and placed in a certain place in the IFC data model. However, the IDM cannot ensure that the areas actually placed in the model are correct.
7 Business Models

To produce reliable standardized computable rules demands good business models. Today, open international standards of high quality are produced all the time, within all kinds of business areas. Similar models should be adapted for the standardized computable rules as well. There is no need to reinvent the wheel, but it is important to recognize what resource demands exist in order to produce high quality standardized computable rules.

Some key aspects to consider are:

- Availability of skill
- Specific cost of standardized computable rule development
- Widespread software support
- Sustainability of the business models (maintainability etc)
- Responsibilities and risk

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
<th>Deliverable</th>
<th>Responsible</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Scope, define the rule source, assumptions and terminology</td>
<td>Document defining sources, scope, assumptions and terminology</td>
<td>Rule Owners</td>
</tr>
<tr>
<td>2</td>
<td>Restructure the rule source according to logic and terminology</td>
<td>Document defining the rule source clearly according to document from first stage</td>
<td>Rule Owners</td>
</tr>
<tr>
<td>3</td>
<td>Create the computable rules, possibly build upon existing rules</td>
<td>Computable rules on standardized format</td>
<td>Rule Owners</td>
</tr>
<tr>
<td>4</td>
<td>Verify the computable rules</td>
<td>Report with verification results</td>
<td>Collaboration with Rule Owners and Software Vendors</td>
</tr>
<tr>
<td>5</td>
<td>Implement the computable rules</td>
<td>Running rule checking software</td>
<td>Software Vendors</td>
</tr>
<tr>
<td>6</td>
<td>Test the computable rules</td>
<td>Test results</td>
<td>Software Vendors</td>
</tr>
<tr>
<td>7</td>
<td>Certify implementations</td>
<td>Certified software</td>
<td>Certification Authorities</td>
</tr>
</tbody>
</table>

Table 6: Responsible Party for each process step
7.1 Availability of Skill

Efficient development of standardized computable rules must involve the access to the correct skill set. If the process of developing such computable rules becomes too complicated, too involved and specialized, the skill set required to produce the rules will become sparse and expensive.

It is thus important for the overall business environment, or “eco-system” of the computable rules creation processes, that as much as possible readily available skills today are used efficiently. For example, developing a new formal language for computable rules creation, might be the best option seen from a pure technical point of view. However, from a business point of view, if good enough formal language with readily available experts exist, it will provide a better option all over due to better access to less expensive skills.

Too complex technology may result in the technology never reaching mainstream. Early developments of Artificial Intelligence (AI) is an example of this.

One such dilemma is with the deliverable suggested for Step 3 in our development process. As mentioned in Error: Reference source not found Error: Reference source not found, we suggest the usage of the IFC Constraint model, delivered as IFC-XML. There is a risk that this format is too complex, and thus too costly to implement, that it will be necessary to find a format where skills are more readily available.

7.2 Cost of Rule Development

It is important to consider the total cost of computable rule development. Not only the actual development cost, but also the cost associated with the potential loss of sale of the underlaying original standard. Or perhaps the underlaying original standard will increase its sale due to the wider availability of the standardized computable rule in software?

Is it possible to combine the cost of maintenance of the original standard, with the cost of maintenance of the computable rule, or are they two separate processes that add cost to the maintenance of any standard?

Clearly, adding additional work to an existing process, also add cost to this process. However, will the added value of the standard make up for the added cost? Or will the synergies between the original rule source and its computable rule counterpart make up for the additional cost?

There is also an initial cost associated with standardized computable rules development. For example, developing a standardized computable rules when rule libraries are readily available, tools to develop rules are plentiful and the whole eco system surrounding rule development provides the availability of
plenty of expertise, is much cheaper than doing it from scratch today.

### 7.3 Software Support

In order to keep the cost down it is important to develop standardized computable rules that can readily be implemented in existing software. If the computable rules depend on non-standard language, solutions or formal procedures, software will be slow at implementing support.

Furthermore, the more widespread the support in software for computable rules, the easier accessible it is to users in the market, and the more widespread its usage is. This will in turn make the computable rule more valuable.

It is also tempting to assume that a complex rule source, which will demand that a user will read and understand the text, before it can be applied to a problem, is less available than the same source available in software, where the only demand on the user may be a push of a button.

### 7.4 Sustainability

The business models need to be sustainable, in the sense that the cost of developing and maintaining a standardized computable rule must be regained, somehow, for the developers of the computable rule.

The complete life cycle of the standardized computable rules must be considered.

### 7.5 Responsibilities and Risk

It is important to clearly define the responsibilities and risk involved in the development and maintenance of standardized computable rules. Table 6 list the recommended responsible parties for each process step. This should be further refined as implementation of the process steps takes place.

In general risk relates to using resources without knowing if these resources can be regained later. Knowing that there is a need for a certain standardized computable rule will lower the risk of developing this, both for rule owners and software vendors.

### 7.6 Possible Business Models

Below is a set of suggested business models. It is also possible to consider a combination of these models as well.
7.6.1 Software Vendors Pay for Certification

Software vendors pay for the certification of their implementation of the computable rules. Certification authorities pay the computable rule owners, who pays for the development and maintenance of the computable rules, for each certificate issued. Software vendors regain this additional cost from increased sales and higher price of software, due to the added value in the certified computable rule implementation.

7.6.2 Rule Owners Pay

The rule owners added value of having readily available standardized computable rules in widespread software in the market, is enough to cover the cost of creating and maintaining standardized computable rules.

7.6.3 Subscription Based

Every time a computable rule is used in software, the user pay the rule owner a small fee. This could be a monthly subscription for unlimited usage, or a per usage fee. Administration of these options can add cost themselves, but more and more of these solutions exist in the market today, so such an adaption should be sustainable.

7.6.4 End Users Pay for Software

When end user buys software that has implemented a standardized computable rule, and extra license fee is added on top of the software fee itself, to pay for the computable rule usage to the rule owners.

7.6.5 End Users Pay for End Result

Instead of paying for software, it is possible to consider scenarios where end users pay for the result of executing software. This is a more targeted system, where the end user only pay for the results a service or software delivers.

7.6.6 Public Authorities Pays

There are business models in place today, on national standards, where public authorities pays the standardization organization for the cost of the standard, but gives the standard away for free to the public. The same scheme can be applied to standardized computable rules as well.
8 Further Development

8.1 Overall Project Recommendations

Having standardized computable rules will lift the efficiency and quality of the design process substantially. Not only will the quality of the produced output from the design process increase dramatically, but the process itself will shift towards a more integrated approach, increasing efficiency and cutting costs. More and more demanding requirements on buildings, like higher environmentally motivated demands, stricter safety rules, better quality of living, etc, enforces more complexity in buildings and thus more complex data models. Manual checking of all requirements will be impossible and data aided rule checking is a natural solution to this challenge.

Software Vendors involved in this project has expressed a clear interest in the work, and agree to the need for standardized computable rules.

This report has the following recommendations:

- Continue with a main project to develop the process suggested here further and conclude on technology and standards to be used
- Design the main project to be iterative, so that small steps can be taken, and then corrected along the way, to ensure the best results for a complex field

8.1.1 Process Recommendations

We have outlined a recommended process in 3.5 Recommended Process Steps. However, it is important to actually implement the process and test it while developing a real world standardized computable rule. In order to achieve this, all actors mentioned in Table 4 must be involved: Rule source owners, experts on the specific rule source, experts on computable rules, certification authorities, software vendors and end users.

The process must be designed to ensure the following:

- Data quality (using IDMds)
- Transparency and documentation
- Iteration based (start with a narrow scope, widen gradually)
- Proper use of existing computable rule libraries
- Separate data quality checking from rule checking
- Separate rule making from rule checking
The stakeholders must ensure that these process steps capture the essence of the rule sources and fit within their current business environment.

### 8.1.2 Technology Recommendations

We have suggested the following technology to be used in the standardized computable rule development process:

- Human readable documents, preferably online hyperlinked documentation
- Structured rule sources in XML, schema to be decided through actual standardized computable rule development
- Standardized computable rules in IFC-XML, based on IFC Constraint model, possibly in combination with semantic technology (RDF, OWL, etc)
- Rely heavily on IFD Library for semantic terminology

The following issues must be resolved:

- What schema for structured rule sources will capture the needs?
- How to deal with geometrical analysis of the BIM, and the related computable rules?
- Is the availability of IFC Constraint skills good enough?
- How to capture necessary human interaction?
- How to develop a good system of cross referencing, reuse and library maintenance for standardized computable rules with this technology?
- How to ensure support for rule checking on generic BIMs and more product specific BIMs at the same time?

### 8.1.3 Business Recommendations

We listed various business models in 7.6 Possible Business Models. It will be necessary to clarify which specific model are best suited for standardized computable rule development and usage. The stakeholders must do so in collaboration with the rest of the market, and clarify roles and responsibilities in a sustainable and transparent way.

It is also important to clarify what the stakeholders should not do:

- Certify software
- Develop software
- Decide what business models work for others
9 Conclusion

There is a window of opportunity to assemble the international community working in this field to establish a standardized computable rule development process. The field itself is also mature enough, but at the same time not yet developed too far, indicating good timing at this point.

Software Vendors have expressed interest, indicating a need in the marked. More complex rules and regulations (environment, energy, safety, etc), will force the uptake of computable rules, as it will be impossible to build without.

Developing standardized computable rules requires more than just technology, collaboration between rule experts and computer experts is necessary, demanding a process approach to the solution, like outlined here.
References


Bell H., Bjørkhaug L., Rønning J. 2004, ICT-Platform for Object Oriented Knowledge in the Building and Construction Industry, B4E, Maastricht, the Netherlands

Bell H. et al. 2005, Providing Life-cycle planning services on IFC/IFD/IFG platform, 10dbmc, Lyon, France

Bjørkhaug L. 2003b, Use of Building Product Models and Reference Data Libraries for project and quality management, CPMS, São Paulo, Brazil


Acknowledgements

We would like to thank the project partners for funding this report. The contributions from the Software Vendors has been essential, and we would especially like to mention: Robin Droegemuller, Heikki Kulusjarvi, Nick Nisbet, Yngve Holte Olsen, Per-Olav Opdahl and Jorulv Rangnes for their contributions to this report.
1 Appendix/Appendix

1.1 Methodology

This part is mostly based on the paper “Foundation for development of computable rules” written by Eilif Hjelseth and presented at the CIB-W78 International Conference at Istanbul Technical University 1st – 3rd October 2009.

Executing of a rule must be included in a knowledge system. Figure 5 is representing a knowledge system and illustrates the relations between a “Knowledge model”, who is represented by the codes (source of rules), “Ontology” who is representing the AEC-industry specific and precise definitions (includes systems as taxonomy and classification), and the “Meta-model” representing the expression of the computer interpretable rules.

![Diagram](image)

*Figure 14: Knowledge model – Ontology – Meta-models (Shakeri et.al. 2001)*

There are today no standards or defined methods for development and documentation of rules for implementation into software. In addition to the logic rule, this must include information about the necessary content of relevant information in the model (e.g. represented as an IFC file).
1.2 Theory and Methods Adaptable for Use in the AEC Industry

1.2.1 Theoretical Foundation for Development of Rules

Based on the multi discipline foundation of AI on logic, mathematics, linguistic, philosophy and informatics there are a large number of theories for defining and develop reasoning systems for logic of rules. There should be no need for starting with “empty sheets”

Methods and theories are in nature deductive, and aiming to re-use the same “principles” in many different situations. This approach can be used to develop modules – topology of rule - same structure of rule applied on different construction parts – reuse and modular assembling of rules into a rule set. To discover this pattern of modules, use of metalanguage will be of great help. John F. Sowa (2007) therefore contradicts people who say that metalevel representations are complex and inefficient. For many applications, metalanguage can significantly reduce the complexity, as in the following sentences in controlled English and their translations to an algebraic notation:

- “Every house is a construction” ⇒ (∀x)(house(x) ⊃ construction(x)).
- “House is a subtype of Construction” ⇒ House < Construction

Every operator of any version of logic is a specialization of some word or phrase in natural language: ∃ for there exists, ∀ for every, ∧ for and, ∨ for or, ⊃ for if-then, ~ for not, ◊ for possibly, and □ for necessarily. The metalevel words for talking about logic and deduction are the same words used for the corresponding concepts in natural languages: truth, falsity, reasoning, assumption, conclusion, and proof. This notation makes it computable and suitable for support by ICT systems – and for defining re-usable modules of rules and logic.

Another interesting thing is to be able to describe relationships between topics, and for this the topic map standard provides a construct called the topic association (Pepper, 2000). Topic maps is well defined in ISO 13250 series of standards. This semantic network builds on the concept of conceptual graphs (CGs). They express meaning in a form that is logically precise, humanly readable, and computationally tractable. Conceptual graphs can be translated to predicate calculus and to the Knowledge Interchange Format (KIF) (Delugach, 2006). Rule Interchange Format (RIF) developed by the W3C consortium is as a Resource Description Framework (RDF) schema.

The RDF triples indicates it connection to OWL ontologies, whit all its possibilities and frameworks for further development. (Bruijn, 2005). According to Beetz et. al. (2005) is an OWL notation of IFCs advantages over generic XML schema representation.
1.2.2 Classification in the AEC Industry

One of the benefits by classification is to find the right information base, and its relation to other information. Amor and Xu (2005) state that the amount of useful information available on the web for A/E/C professionals increases inexorably. They have tested numerous search engines allow users to identify potentially useful information in this vast resource, though the majority of these systems work purely on the search terms entered by the user. This means that the web pages which are found are often not as relevant to the user's needs as would be expected. What is returned is certainly far from the promise of the semantic web where the properties of the content can be readily ascertained.

There will therefore still be need for the traditional developed classification system. In a report developed for Standards Norway, Bakkmoen (2009) point out that the complexity and number of different systems clearly illustrates the need for harmonization or mapping between the systems if structure and classification at any time needs to be transferred across borders between nations, organizations, or classification systems. With references to preliminary study he finds that IFD Library appears to be the most obvious alternative available to the building and construction industry.

Borup (2008) points out following fundamentals for classification;

- Existing classification systems like OmniClass, BSAB, Uniclass are Building Information Models of the classified and defined objects in the construction domain. All objects in classification BIMs are corresponding to concepts. They all have names based on the language used as a tool in our thinking
- Classification systems need solid theoretical foundations and standardized methods for their creations -e.g. use of the international standardized concepts and methods in the terminology domain including standardized definitions of definition
- Models muddle in the construction industry can be an increasing problem.

On the other side, it can now be a “Window of opportunity” to establish a sustainable foundation for rule development. The technical side of BIM / IFC is in a large degree solved, and the remaining parts can be solved by extension of entities and/or property sets in the IFC-schema. (Ding et. al. 2004). The knowledge theories and methods itself are well enough developed. Common use of the standardized 3-tier framework in table 1. can be used for specifying deliverables (from different suppliers) within each layer. Instead of the software depended situation of today. Deliverables in second layer could be re-used in a number of implementations in third layer.

1.2.3 Concepts for Mapping Ontology – IFD

IFD – Interntioanal Framework for Dictionaries can be rgardes as a concet for
mapping different classifications. As illustrated in figure 6, IFD is a mapping between different parts of different classifications tables. This will lead to an increasing numbers of relations, and a method for presentation of these relationships is by use of the Hyperbolic tree concept. (Bell and Bjørkhaug, 2006).

In order to automatically verify the information in an exchange process we need to detail the information further than the general level of the IFC standard. For example, when the architect supplies information about the type of materials in the beams and columns, she must do so using a plain text string. Even if she spells this correctly, there is no guarantee that the receiving application will understand exactly what this text string means. And what if she uses a different language, dialect or uses the plural form of the word? Ideally the computer should be able to understand even this type of information in the IFC formatted information received. This is typically the scenario addressed in semantic searches on the web. (Bell et.al., 2008, Bell and Bjørkhaug, 2007b)

The IFD standard is based on the ISO 12006-3:2007 “Building construction -- Organization of information about construction works -- Part 3: Framework for object-oriented information” standard, and has many similarities with the EPISTLE4 standard for the Oil and Gas industry (Bell and Bjørkhaug, 2007a).

1.2.4 Theories and Methods for Finite Domains

But is it still possible to avoid the “Knowledge soup”? One approach can be to
consider the AEC-industry sources for rules (standards and likewise described codes and regulations) as finite domain of knowledge and professional language (ontology).

The first element is look if there is a language where true and common understanding. Alfred Tarski (1935) concludes in “The concept of Truth in Formalized Languages” that it is a hopeless exercise with regard to natural language, because of its complex and mutable nature. Given the non-universal nature of formal languages (specifically, the usual absence therein of terms belonging to the theory of language), a distinction must be made between the object language (the language under study) and the metalanguage. The metalanguage contains the names of the expressions of the object language and of the relations between those expressions, and usually the full vocabulary of the object language. But if one look at finite domains, solutions is possible “The problem of the definition of truth obtains a precise meaning and can be solved in a rigorous way only for those languages whose structure has been exactly specified”. For other languages — thus, for all natural, "spoken" languages — the meaning of the problem is more or less vague, and its solution can have only an approximate character (Tarski, 1944). If we let this be the “indicator “that a solutions is possible within the AEC domain of knowledge, The chaos of Babel-like communication can be avoided or at least reduced with a common language of rules.

### 1.3 Methods for the AEC Industry

#### 1.3.1 From English to Rulish Versions of Standards

Based on theory (Sowa, 2000, 2006, 2007 and Tarski, 1935, 1944) is should be possible to develop a logic system with a finite domain and a structured language. The languages and semantics in standards are written in a defined way, and are suitable for translating into formal notation in a truthful way. The argument for this statement is based on the ISO normative rules for structuring and drafting international standards in Table 7: Requirement (ISO, 2004).
<table>
<thead>
<tr>
<th>Verbal Form</th>
<th>Equivalent Expressions for Use in Exceptional Cases</th>
</tr>
</thead>
</table>
| **shall** | is to  
is required to  
it is required that  
has to  
only ... is permitted  
it is necessary  |
| **shall not** | is not allowed [permitted] [acceptable] [permissible]  
is required to be not  
is required that ... be not  
is not to be |

Do not use “must” as an alternative for “shall”. (This will avoid any confusion between the requirements of a document and external statutory obligations.). Do not use “may not” instead of “shall not” to express a prohibition. To express a direct instruction, for example referring to steps to be taken in a test method, use the imperative mood in English. Example: “Switch on the recorder.”

Table 7: Requirement (ISO, 2004)

- For “Recommendation”, (Table H.2) the ISO standards use the verbal form: Should / should not.
- For “Permission” (Table H.3) the ISO standards use the verbal form: May / need not
- For “Possibility and capability” (Table H.4) the ISO standards use the verbal form: Can / cannot

all with equivalent expressions for use in exceptional cases similar to Table 7 (ISO, 2004).

By use of semantic method it should be possible to developed “Rulish” version ready for implementation into software. Laws and regulations also have a similar way of using modal auxiliary verb. This use of normative reference is also implemented in the BIM-manual version 1.1 from Statsbygg, Norwegian Public Construction and Property Management (same as GSA in USA) (Statsbygg, 2009).

1.3.2 ISO Supported System for Domain Knowledge

According to Sowa (2000) should the general principles for constituting an expert system be based on a background knowledge about the world, including ontology, axioms, and defaults. This includes the following topics:

- **Ontology:** A classification of the types and subtypes of concepts and relations necessary to describe everything in the application domain.
• **Definitions:** Necessary and sufficient conditions that define new types of concepts and relations in terms of more primitive concepts and relations in terms of more primitive types.

• **Constraints:** General principles or axioms that must be true of the instances of those concepts.

• **Defaults:** Information that is expected to be true of the instances of various concept types.

• **Behavior:** Rules that govern the actions by and upon each type of object and the interactions of collections of objects.

For mapping of specification to logic is Conceptual Schema Modeling Facilities (CSMF) useful. The ISO JTC 1/SC 32 project on Conceptual Schema Modeling Facilities (CSMF) is developing standards for appropriate languages and tools. (Sowa, 2000). Conceptual modeling is central for systems analysis, database modeling, and knowledge engineering, to support the development processes from original codes into computable rules.

### 1.3.3 MOKA and Participants

Methodology and software tools Oriented to Knowledge based engineering Applications (MOKA) is a name given to a methodology as a part of ESPRIT research program. It was assumed to work out the following:

• knowledge representation forms of a product and its designing process as well as the methods of its record,

• computer application for aiding record, representation and managing of the knowledge,

• possibilities of further automatic generation of KBE application code from computer application.

MOKA is used for obtaining knowledge in designing process, for elaborating KBE (Knowledge Based Engineering) and for creation of knowledge base for this system. It uses MOKA methodology and in particular informal model from this methodology and the concept of ICARE forms (Illustration, Constraint, Activity, Rule, Entity). Knowledge referring to the structure of a designed product and its designing process is collected by means of forms.

Ontology’s can be created by means of Protégé application and can be exported to many different formats including RDF(S), OWL XML Schema. Protégé is a free open source tool, it is an application which aids creation of knowledge bases, including ontology edition and knowledge acquisition from experts. (MOKA, 2007).

### 1.3.4 Using the IFC Constraint Model

The IFC constraint model can be developed such that both simple and complex
constraints can be captured. This is done through the provision of a constraint aggregation where the aggregation can be characterized by a logical AND, logical OR or logical NOT operator. The relationship between IDM and IFC is illustrated in Table 8: Relationship between IDM and IFC (Nisbet 2008).

<table>
<thead>
<tr>
<th>IDM</th>
<th>IFC Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process Map</td>
<td>IFC Process Model</td>
</tr>
<tr>
<td>Process/Sub Process</td>
<td>IfcRelAggregates/IfcRelNests</td>
</tr>
<tr>
<td>Process/Actor</td>
<td>IfcRelAssignsToActor</td>
</tr>
<tr>
<td>Process Sequencing</td>
<td>IfcRelSequence</td>
</tr>
<tr>
<td>Process Attribute</td>
<td>IfcRelDefinesByProperties/HasPropertySet</td>
</tr>
<tr>
<td>Process Constraint</td>
<td>IfcRelAssociatesConstraint</td>
</tr>
<tr>
<td>Constraint/Sub Constraint</td>
<td>IfcConstraintAggregationRelationship</td>
</tr>
<tr>
<td>Model View</td>
<td>IFC Constraint Model</td>
</tr>
</tbody>
</table>

*Table 8: Relationship between IDM and IFC (Nisbet 2008)*

The IFC Constraint concept can be integrated into the IDM development method described in ISO/DIS 29481-1 “Building information models -- Information delivery manual -- Part 1: Methodology and format”. Rules that are applied to functional parts and exchange requirements are collected together into rule-sets. Each rule-set is expected to deal with a particular topic. However, a rule-set may contain rules from many origins provided that they are collected together in an organised way. (Wix et. al., 2008).

### 1.3.5 Translation Versus Transformation

If it is possible to get at direct match from original languish into “rulish”; the rule definition process can be performed as a translation. However, not all code related text, even standards is suitable for this. The first step is trying to “normalize” the original text into a formal text that can be translated. If this will result in – or can result in – different consequences than the original code, one must be utmost careful. However, automatic rule checking will in many cases be very useful. A way to go around is to explicit define this as a transformed rule, and, a dialogue box give information about this when the rule is activated in the rule checker software.

### 1.3.6 Official Computable Rule-sets - Rulish Version of Standards

As discussed above there can be minor formal or theoretical differences between the standard in native languish and “rulish” without this leads to any general consequences for assessment of the building. An example on this can be the
accessibility for a wheelchair through doors. The ISO/DIS 21542 “Building
construction - Accessibility and usability of built environment” defines a
minimum width of light opening to be 800 mm. For assessment of this one must
have the very detailed information about the door-casing – and the opening angel
of the door. If the door is only open in 90 the thickness of the door lead will cover
some of the opening compared to complete open parallel to the wall is stand in.
This demands very much information about the door and it instance (opening).
Knowing that doors are industrially produced in 100 mm intervals (Modules =
M), using a 9 M door (outer door-casing width 888 mm) will give adequate
opening width. Without this transformation, the accessibility would be very hard
to check automatically.

By having an official “Rulish standard” version, these transformations could be
transparent, and solutions based on consensus can be applied instead of “tricks”
from the software developer. In the example above the Rulish standard should
have a warning (information) for door with less than 950 mm outer door-casing
width, that the defined properties to the door must be manually checked.

1.3.7 Example of Projects Using a Formal
Methodology

SMARTcodes (see 2.3.1 SMARTcodes) have applied some form of formal
methodology, using tools to help the domain expert clarify her implicit rules in
the rule source by coloring certain textual parts. Then these are parsed by a
computer to generate computable code (IfcConstraint models).

The SWOP -PMO project (see Error: Reference source not found Error:
Reference source not found) on the other hand is using semantic web formats
like OWL/RDF to represent the knowledge. Then SPARQL is used for querying
and Rule Interchange Format RIF to represent the rules. The RDF/OWL
representation is not derived from written knowledge but have to be remodeled
in accordance with the rules of OWL/RDF.

References – Annex/Appendix

Bakkmoen, K.I., 2009. Classification of building and civil engineering types for
the AEC industry Report developed for Standards Norway, C. F. Møller
Architects, Norway.

Bell, H. and Bjørkhaug, L. 2006. A buildingSMART Ontology eWork and

Bell, H. and Bjørkhaug, L. 2007a. IFD and IFC, http://dev.ifd-
library.org/index.php/Ifd:Ifd_and_Ifd

Bell, H. and Bjørkhaug, L. 2007b. IFD in a Nutshell, http://dev.ifd-


Sprachen’, Studia Philosophica I.
2 Experience From Software Vendors

2.1 Experience from AEC3

AEC3 develops rules-based approaches for the construction industry. It was closely involved in the Singapore ePlanCheck solution and subsequently has developed these ideas for the UK Health and Safety Executive, the Scottish building regulations and others. In 2006 AEC3 created the SMARTcodes methodology. The initial demonstration was held in July 2006, and the first full public demonstration in November 2006. The first implementations were on three systems:

- Singapore CORENET ePlanCheck with Fornax
- Solibri Model Checking with SMARTcode extension
- AEC3 XABIO using EDM and Octaga

Since then, the methodology has been developed and validated, and a fourth platform added:

- AEC3 Compliance1 command line application with exchangeable dictionaries.

SMARTcodes has several key features:

- User-created markup of actual Code. This empowers problem-owners to prioritize their content, and manage versions and updates. Codes can be partially or fully marked-up, whether or not methods exist for checking every concept needed.

- Automatic derivation of the logical constraint model. This is in effect the semi-compiled version of the code which retains its links back into the code.

- Full implementation of three-value logic (true, false, unknown). This ensures that even when some information or methods are not available, the checking engine can create useful results and advice. IDM methods have a role in improving the quality and completeness of building models, and the development of SMARTcodes feeds naturally into this: AEC3 has shown that exchange requirements can be documented automatically from SMARTcodes themselves, and that supplementary information can be gathered and incorporated automatically.

- Separation of the logical engine from the dictionary of methods. The
checking engine is conceptually simple and implementations are verifiable. Individual Codes may need a small or large number of atomic concepts, but experience suggests that these atomic concepts can be re-used. For example, testing if a fabric component is part of the building envelope is used by most Codes, even if the requirements and exceptions created using this atom vary widely.

- Multiple checking platforms. The logical model is delivered as an IFC or ifcXML model using the iFC constraint model. This representation was chosen to be the same as the likely representation of the building model. This has ensured that all checking platforms have been able to implement SMARTcodes easily. In the case of ePlanCheck, the SMARTcode approach was easily inserted into an existing framework.

- Issues and reporting related to source text. Different checking engines have adopted differing approaches to reporting:
  - Reporting and viewing within the checking application
  - XHTML web pages of issues, with an embedded 3d viewing component
  - PDF reports with static images.
  - Proprietary XML based reporting of issues.
  - IFC and ifcXML models containing IfcApproval objects, for onwards management or presentation using style-sheets. AEC3 believes that this approach is the way forward, as it will integrate with improved ‘Request For Information’ (RFIs) and Issue Tracking processes.

SMARTcodes aims to clarify the multi-disciplinary roles: Code (not coding) experts perform the mark-up and enhance the dictionary if necessary. Building modeling experts review these terms to identify the model data requirements. Software coding experts may contribute the implementation of detailed algorithms, such as finding the shortest escape route. Where Codes refer to standard external methods (such as ASHRAE 90.1) then these may run as a pre-process prior to code checking, inserting their results into the building model.

The key areas for future development is around the role of the dictionary. Initially, the SMARTcode markup tools allowed the user to use any terminology. The checking engine was configured to respond to all the terms used. Given the variety of spellings and phrases, a dictionary was introduced. This was prepared in as a spreadsheet and associated key object topics with relevant properties. This list was used to guide the development of the catalogue of methods. Transformation tools were used to represent this simple dictionary in RDF/OWL form, and this representation was used to constrain the mark-up.

The AEC3 Compliance1 checking platform allows third parties to create and manage their own dictionaries of methods, as distinct DLLs serving the core application. The first application of this architecture was for security analysis for
the US Department of State Overseas Building Office.

The next development will be to standardize the dictionary around the IFD. Code authors will be able to develop their dictionary locally, using an XML representation of the IFD. IFC methods and services can then be registered as alternate language representations. Once stabilized the dictionary can be incorporated into the central IFD server.

### 2.1.1 Process Experience

The following is our rule making process experience:

2. Verify (1.): peer review of summary reports and reverse engineering.
3. Implement some new atomic concepts.
   1. Simple extraction or deduction from model
   2. Advanced algorithms
4. Test each of (3.) independently.
5. Use.

Finally, we end with the following realizations:

1. Automatic Code Compliance puts BIM in front of a larger and less receptive audience than usual
2. Overcoming the belief that code models, dictionaries and building models have to be perfect/complete/specially-made.

### 2.2 Experience from Jotne EPM Technology

It is of uttermost importance that we isolate problems in data from problems in the rule checking method. The rule checking method is very often "easy and straightforward" as long as you have accurate and reliable data. The last topic can only be solved if we have a good specification of process & exchange requirements. If we bring data checking into the rule-checkers, we get a very messy and proprietary solution, if we are able to first validate data with standardized exchange requirements error messages, we get leaner, better, reduced development time and re-usable checking methods.

Rule Checking should be integrated with the BIM Manual development, e.g. the BIM Manual should be a computer-sensible and human readable document designed for direct implementation/configuration and the checking methods should be available from a library.
Rule Checking should be closely linked with the IDM work and rule checking methods should be standardized within ISO TC59 / SC13.

So, if you run a BIM Manual Process (similar to the one we just have completed for Statsbygg Vestbanen www.statsbyggcompetition.no), you can easily add/remove checks dependent on demand, e.g. deliver spaces, check space program, check space collision/overlap, check functional area efficiency, check early-stage energy performance, etc. Or you can skip some of the tests if that is what you want to do.

### 2.3 Experience from Solibri

The model checking process should always begin with general checks to verify that required building components are available and that there are no major issues with the model. Once this is done, more advanced analysis like building code checking can be conducted, which will generate consistent and reliable results. As an example, egress rules should be executed only when you have spatial components defined correctly and doors in place.

To make this happen, Architects and Engineers need to know in the first place what information is needed in the BIM files. This can be done by having clear BIM guides like e.g. Statsbygg (Norway), General Services Administration (USA), Senate Properties (Finland), have been doing. These guidelines will tell what information is relevant at which phase of the design. Also, designers need to have good knowledge how their choice of BIM authoring tool should be used in order to produce required information. In practical projects it happens quite often that architects are using tools in their BIM authoring software that produce good documents utilizing for example parametrized objects. Sometimes this leads into situations where the 2D document (produced from the BIM file) has bathroom space with surrounding walls, doors and windows, sinks etc. visible, but the BIM file only contains one bathroom object, without any information of the space, walls, door etc. This is why it is important to have local vendor specific instructions how a certain BIM authoring tool should be used to comply with the local BIM guide.

Another aspect to consider is the phasing of the model checking. The current practice of running building code checking with complete design leads to expensive rework in case part of the design fails the checking. Instead of waiting for a complete BIM models to be handed over to building code checking, we could utilize the full potential of BIM process by doing checks earlier when relevant information is available.

The architects and engineers can be guided to check their models early and often. The checking should be done before a design is handed over to other disciplines. Checking early and often will also increase the possibility that the final checking by building authorities runs smoothly without surprises and the actual building can start earlier.
Area calculation/measurement is a good example of this. If we wait until the architectural design is ready for permitting, the architectural design is most likely already been distributed to other designers and they have started their work based on spaces defined in the model. Now during the area measurement we notice that some of the spaces exceed the limits and changes need to be done. This will effect how the design is done by other disciplines. If instead this check would have been done during the very early spatial design, maybe having only the space objects in the model, this problem could have been noticed earlier.

This is just one example but there are plenty of similar issues that when noticed early during the design process would clearly save time and energy for all parties. This is why it is recommended that we study automated building code checking as a two stage or two role process. One being an early phase “self check” by the architects themselves based on the information available and what of the building code checks can be performed during the design. The second phase would be the official building code checking for permitting when design for this phase is completed.

2.4 Experience from SINTEF Building and Infrastructure

The biggest challenge lies in catching the knowledge about a rule from an expert and into a system the computer can utilize. In order to make this happen, several systems must be in place:
• Some kind of easily expressive language the expert (domain expert, usually not a computer literate) can express their knowledge in
• Some kind of contextual building blocks, upon which the domain expert build their rules
• Ability to reference, and reuse previously built knowledge bits, or rules, in newer ones

Today, all of these systems are in place for written materials:
• Norwegian (or English) is used by the domain expert to express their knowledge, or rule
• A vocabulary for their domain exist and well known examples can be applied
• Simple references are frequently used to show dependence and likeness to other knowledge bits or rules

It may turn out that this is the best form for an expert to express his knowledge or rules, while a computer, augmented with semantic technology, and continuously build better conversion technology, to make the textual form computable.