EXPLORING SEMANTIC BASED MODEL CHECKING

Eilif Hjelseth, Ph.D. student, eilif.hjelseth@umb.no
Norwegian University of Life Sciences (UMB), Dept. of Mathematical Sciences and Technology, Norway
Nick Nisbet, Director, nn@aec3.com
AEC3 UK Ltd, United Kingdom

ABSTRACT

This paper explores the foundation of semantic based model checking concepts. Development of computable rules in a pure semantic based concept is characterized by "soft coding" by following a pre-defined mark-up methodology for linguistic (text and numbers) analysis, organization, execution and reporting. The software programming for this can be done automatically or semi-automatically based on predefined procedures. This enables a person skilled in the AEC domain to develop applicable rules without support of programmers. The rules can be then be applied to the semantic content of a Building Information Model, typically in the IFC format. Whether it is possible to develop a valid and reliable system applicable to rule sources (laws, codes, regulations and standards) depends on testing two key hypotheses:

The user's perspective is basis for the first hypothesis which says that a pre-defined semantic system or toolset can be used by person skilled in the AEC domain (not software programmer) to define rules. The second hypothesis proposes that a system for automatic, or semi-automatic, generation of applicable rule sets for software implementation can be developed.

Keywords: Model checking, semantic, BIM, IFC

1. INTRODUCTION

The facilities industry is highly regulated by a large number rules given by public laws, codes, and regulative, standards national and international levels which taken together identify what is permitted or recommended. User demands can also be considered as the same as rules. Rules can be complicated in itself and interlinked with other regulations or preconditions. This situation indicates a need for product model checking systems that can support the design process through into the entire facility life cycle. The question is if and how such a system can be designed to fulfill demands for time- and cost effective development and implementation into software. This paper will show that a semantic system based on four mark-up operators can provide a trustworthy result. Due to logic consistency, we do also want to show that it is possible to auto generate software code from the marked up text.

2. MODEL CHECKING

We want to introduce automatic model checking tool to support the design processes. The term Model checking is used as a term for checking content of information in relation to defined requirements. Code compliance checking is classified as specific case where model checking "validates" the design. (Hjelseth and Nisbet, 2010).The challenge is how to do this in the best way overall. The obvious way is to have a procedure where an equally skilled person checks the design of the model, with or without some guidelines or checklist. This manually system is limited by the availability of such skills, and is error prone during repeated checking. Every checking exercise has relative high cost and takes time. Another solution can be replacing the expert with an expert systems based on AI (artificial intelligence) or KBE (knowledge based engineering) systems. These systems have large potential – but the cost and time for developing these systems dilute the benefit. An example of this is given by John F.

Sowa (2006) who refer to the Halo project where representing the knowledge in a chemistry book into an AI system was tried. The results were a score from 40% to 47 % correct and a cost of about $ 10.000 per page textbook. One explanation was the heterogeneity of the chemistry text leading to the “knowledge soup”. The “knowledge soup” arose for four reasons: a) Overgeneralizations, b) Incomplete definitions, c) Conflicting defaults and d) Unanticipated applications. Sowa further notice that experience shows that these exceptions and borderline cases result from the nature of the world, not from language or logic (Sowa, 2000).

3. SEMANTIC SYSTEMS

The focus on semantic systems is very high related to knowledge search. On the internet this new semantic approach is often called WEB 2.0. The semantic approach is also suitable for model checking, which is a kind of knowledge search supported by rules (also called methods, procedures, algorithms, formula and so on). By using a semantic approach we are able to handle two aspects related to:
a) Identification of the computable rules by a person with AEC domain skills
b) Implementation of applicable software code based on common predefined measures.

Regarding the first point, previously the development of rule sets for rule checking software has been vendor lead. This gives a “Black-box” solution and requires much testing prior to relying on the result. By empowering the domain experts we get a more transparent and direct method and can come much closer to the knowledge the rules are based on, reflecting the context definitions. In short, meaning is dependent on many factors, not just one. The correct interpretations of what might be understood, is obtained by using the viewers perspective. (Henderson (2010). An important aspect of checking in the AEC industry is that what is “checked” is very related and interacted with it “surroundings”. The source for rules (laws, codes, regulations, standards and so on) is also interlinked with each other. An AEC professional is far more skilled to appreciate this than a software developer.

The semantic ambiguity in the reference data complexity is decreased by use of;
- Dictionary; vocabulary, terms and definitions,
- Taxonomy; classes in sub-/superclass hierarchy,
- Ontology; constraints and connections.

Ideally, the vocabulary used by the domain experts would be drawn from an existing ontology. At this early stage, the process is being reversed and key concepts and properties are being identified during the tagging process. Lists of synonyms and shades of meaning that depend on context are being catalogued.

4. FOUNDATION FOR A MARK-UP LANGUAGE

Based on semantic theory (Sowa, 2000, 2006, 2007 and Tarski, 1935, 1944) it 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. An example is the ISO normative rules for structuring and drafting international standards in table 1.

Table. 1: Requirement, ISO Table H1 (ISO, 2004)

Verbal form	**Equivalent expressions for use in exceptional cases**
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."	

For "*Recommendation*, (ISO Table H.2) the ISO standards use the verbal form: *Should / should not*, for "Permission" (ISO Table H.3) the ISO standards use the verbal form: *May / need not*, and for "*Possibility and capability*" (ISO Table H.4) the ISO standards use the verbal form: *Can / cannot*, all with equivalent expressions for use in exceptional cases similar to Table H.1. (ISO, 2004). By use of semantic method it should be possible to develop a rule based version ("Rulish" version) ready for implementation into software. Laws and regulations also have a similar way of using modal auxiliary verb.

5. DEVELOPMENT OF SEMANTIC BASED RULES

Based on the theoretical foundation described below, there should be possible to develop a semantic based system for model checking who gives trustworthy results for use in the AEC knowledge domain. A semantic solution 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

Legislation and regulations typically present as apparently well-structured documents. Irrespective of the relative complexity of any particular document, it is possible and useful to identify the common constructs being described. It is proposed that regulations can be broken down into five fundamental concepts. These five concepts were elected to be most familiar to regulatory experts and designers, rather than to application or database analysts. The most general of these has been named the 'check' and typically demarcates a section of the regulation that is distinct and independent of any other. 'Checks' are often, but not necessarily, closely related to the named or titled sections in the document. It is a characteristic of regulations that every 'check' must be in some way satisfied.

A 'check' is not an indivisible (atomic) concept: it can be analyzed down further into four subsidiary constructs. The most obvious and most easily identified are the 'requirements' as these are associated with the future imperatives 'shall' or 'shall not', it is required that a check contains at least one 'requirement'. Secondly, there will be text that identifies the 'applicability' of the check. These are often compounded, for example 'external windows' which compounds the 'external envelope' concept with the 'window' concept. These phrases need not relate directly to the topic of the regulation or the topic of the overall check. For example, if a check applies in 'seismic zone X', this is a property of the building site, not of the structural integrity of a particular building component. In general, there will be one or more phrases defining the applicability. One similar but distinct case is where a 'selection' of alternative subjects is offered, for example 'doors, windows and other openings'. Lastly, there may be one or more 'exceptions'. These are the opposite of 'applicabilities', and conversely work by

exclusion. To summarize, a regulation contains a number of ‘checks’, and each check contains a number of ‘requirements’, ‘applicabilities’, ‘selections’ and ‘exceptions’.

Actual checks may contain a number of requirements, applicabilities, selections and exceptions. It is important to identify how these combine. Requirements are typically cumulative; if there are several stated requirements, then it is to be expected that all must be satisfied. Similarly, applicabilities are cumulative and all must be met. However, if there are many exceptions or selections, then typically these are alternatives, and only one will be relevant. Using a short notation:

Requirement $R_0 = R_1$ *and* R_2 *and* R_3 *and*… R_n
Applicability $A_0 = A_1$ *and* A_2 *and* A_3 *and*… A_n
Selection $S_0 = S_1$ *or* S_2 *or* S_3 *or*… S_n
Exception $E_0 = E_1$ *or* E_2 *or* E_3 *or*… E_n

Sometimes more complex linguistic structures are encountered, usually reflecting more complex logical intentions. Methods for representing these have been developed, using subtypes of the four concepts. However complex any actual examples prove, standard logical calculus allows these to be expanded and manipulated, as shown in Figure 1.

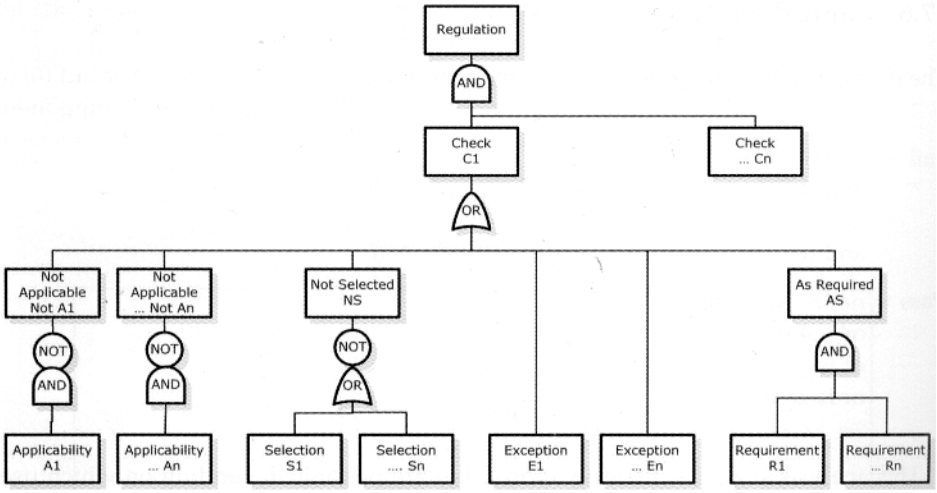

Figure 1: Applications use a logical expansion of the mark-up hierarchy.

The check, (C_1), is then equal to a logical combination of the ways of passing the check, which can also be summarized in a short notation.

$C_0 = R_0$ or not A_0 or not S_0 or E_0

The regulation is the logical combination of the distinct checks, which can also be summarized in a short notation. The argument is made more general by observing that most check clauses function as requirements.

$Regulation_0 = C_1$ and C_2 and C_3 and … C_n

Regulations can be grouped into rule-sets, representing a domain (field / area) which is to be checked. An example is accessibility for electrical or manual wheelchair.

6. SUGGESTION FOR A MARK-UP LANGUAGE

The language in a mark-up should be based on as few operators as possible, and must also used in a well documented and transparent way. Our suggestion is to use following operators; select, applies, requirement and exception. Applied on a text, the user highlights any phrase that means:

- more scope as a ‘select’
- less scope as an ‘applies’
- ‘shall’/’must’ etc as a ‘requirement’, (including alternative requirements)

- ‘unless’ etc as an ‘exception’, (including composite exceptions).

The relation between the operators and the original building codes in text is made apparent by a color system according to the mark-up language. An example of the mark-up operators and its related colors is software is illustrated in figure 2. The colors were chosen for acceptability for those with visual impairments.

			
Requirement {blue}	Applies {green}	Select {*red*}	Exception {*orange*}

Figure 2: The four operators for rule development

Those four operators used for marking up text can be visualized by colors related to the operators, e.g. blue, green, red and orange. This gives the user an instant overview of what and how the rules are structured. The naming of the operator is specially chosen to correspond with the way standards, codes, regulative are written, which reflects natural language.

Requirement (R), Applicabilities (A), Selection (S) and Exceptions (E) constructs can be identically attributed to have a *topic*, a *property*, a *comparator* and a *target value*. The topic and property will ideally be drawn from a restricted dictionary composed of terms defined within the regulation and normal practice. The value (with any unit) may be numeric, whereupon the comparators will include ‘greater’, ‘lesser’, ‘equal’ and their converses. If the value is descriptive, then only the ‘equal’ or ‘not equal’ comparators are relevant. If the value represents a set of objects, then the comparator may be any of the set comparison operators such as ‘includes’, ‘excludes’.

7. EXAMPLE OF IMPLEMENTATION

The ICC SMART codes project can be used as an example of how this semantic concept can be implemented into software and applicable model checking. The components of a regulation are illustrated in table 2 by a rule on moisture control. The rule ‘applies’ on building elements. Based on the ontology, this also includes walls, windows, doors, ceilings, slabs and so on. Under ‘selected’ is the building element that has to be present. Without a wall, moisture control is not possible. Under ‘except’ is a number of conditions where this rule will not run, for example moisture is not a problem in defined areas of the country classified as climatic zone 1, 2 or 3. If the building elements are moisture proof, frost proof or condensation proof, moisture is of course no problem for the building elements. The resulting metrics are summarized in table 3.

Table 2: Example clause (based on ICC IECC 2006 502.5 Moisture control. (Mandatory)).

Rule source	**ICC IECC 2006 502.5 Moisture control**
Rule description	All {*green/*}framed{*green*}{*red/*} walls, floors{*red*} and {*red/*}ceilings{*red*} {*orange/*}not ventilated{*orange*} to allow moisture to escape shall be provided with an {*blue/*}approved vapor retarder{*blue*} having {*blue/*}a permeance rating of 1 perm{*blue*} (5.7 × 10 –11 kg/Pa s m2) or less, when tested in accordance with the desiccant method using Procedure A of ASTM E 96. The vapor retarder shall be {*blue/*}installed on the warm-in-winter side{*blue*} of the insulation. Exceptions: {*orange/*}Buildings located in Climate Zones 1 through 3{*orange*} as indicated in Figure 301.1 and Table 301.1. In construction where {*orange/*}moisture{*orange*} or its {*orange/*}freezing{*orange*} will not damage the materials. Where other approved means to avoid {*orange/*}condensation{*orange*} in unventilated framed wall, floor, roof and ceiling cavities.

Table 3: Overview of mark-up in a rule for moisture control

Mark-up operator	Mark-up color	Identification of construction object	Property of object	Logic relation	Value
apply	*green*	building element	construction	=	framed
select	*red*	wall	*(existence)*	=	*(true)*
select	*red*	floor	*(existence)*	=	*(true)*
select	*red*	ceiling	*(existence)*	=	*(true)*
except	*orange*	building element	ventilated	=	*(true)*
except	*orange*	site	zone	=	1
except	*orange*	site	zone	=	2
except	*orange*	site	zone	=	3
except	*orange*	building element	moisture proof	=	*(true)*
except	*orange*	building element	frost proof	=	*(true)*
except	*orange*	building element	condensation proof	=	*(true)*
require	*blue*	building element. vapor retarder	*(existence)*	=	*(true)*
require	*blue*	building element. vapor retarder	permeance	<	1
require	*blue*	building element. vapor retarder	location	=	warm-in-winter

8. THE LOGIC OF SEMANTIC

There is no lack of sophisticated AI and expert systems. Some can even beat the world champion in chess, as when Gary Kasparov was beaten by IBM's Deep Blue in 1997 (Russell and Norvig 2010) However, the is a lack of time- and cost effective systems for development of computable rules for practical use. The challenge is within Einstein's famous quote: Make is as simple as possible, but not simpler. In our opinion the semantic approach does offer this change of paradigm. As we see form Peirce's truth tables for Boolean algebra where he introduce the if-then table. He also introduce the inclusive or instead of Boole's exclusive or. Peirce preferred to use the connected symbol –< to indicate that this is a single indivisible operation and not a combination of < and = .The different combinations are set up in figure 3 as Peirce's truth table for Boolean algebra (Sowa, 2000).

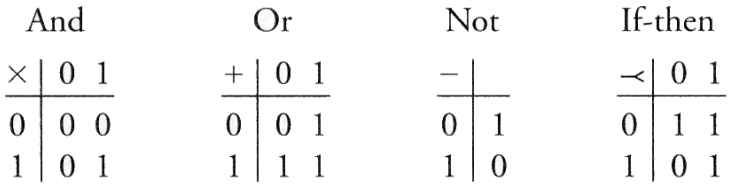

Figure 3. Peirce's truth table for Boolean algebra (Sowa, 2000)

Both Boolean algebra and semantic algebra have basis in natural language, but with some differences. The semantic approach uses four operators; **R**equirement, **A**pplies, **S**elect, **E**xception (RASE) to identify the natural concepts. Boolean uses three operators And, Or , Not (AON) to highlight the written words.

The "RASE" approach is analogous to Aristotle's analysis in 'On Interpretation' of normative sentences, where all sentences are classified as Universal or Particular, Affirmative or Negative (table 4) . Both grids organize normative statements, the difference being that Aristotle organized the grid by how they are expressed, we have organized them by how they combine;

Table 4: Aristotle's normative sentences (Sowa, 2000)

	Affirmation	**Denial**
Universal	Every A is B	No A is B
Particular	Some A is B	Not every A is B

RASE can be logically transformed to AON, and vice versa. Table 5 is a matrix of the logical conjunctions for RASE, which confirms that there are exactly four concepts needed.

Table 5: Relation between semantic and Boolean operators

	-	**not**
and	R	A
or	E	S

For implementation, the use of the four semantic operators was preferred to having to introduce two operators, one list-based, one unary based, as required by AON. All four RASE logical operators act on a list of metrics. This makes the code to process them simple and consistent, even where individual metrics are allowed to have an UNKNOWN value as an alternative to TRUE or FALSE.

Experiences from a number of projects (ICC and GSA in USA, Byggforsk Knowledge systems in Norway) are that the operators correspond to our thinking (cognition) and support organizing of texts. Having a transparent interpretation makes it much easier to detect errors. This gives a more trustworthy result than a more sophisticated and generic system. The possibility for automatic generation for applicable software code is indicated by the relation to the IFC constraint model.

9. AUTOMATIC GENERATION OF APPLICABLE SOFTWARE CODE

The development of computable rules is enabled through a predefined semantic system. It will therefore be possible to make automatic generation of applicable software code. There is a 1:1 relation between the semantic operators and logical operators in software code identified by the mark-up. The markup XML is inserted into the original text.. The marked up document is then processed by a single recursive algorithm to convert this into a single computable rule that can be expressed in the IFC Constraint schema. The IFC Constraint schema is in effect a standardized computable rule format, usable by any rule engine capable of reading an IFC building model file.

10. USE OF THE LOGICAL CONSTRAINT MODEL

Alongside computable rules , the model to be checked must contain relevant information. The Building Information Model (BIM) is in practice represented by an IFC file. The IFC file format (IFC schema) is defined in the ISO standard ISO 16759 and contains over 700 objects. However, there are today some limitations in the scope of implementation of information capture in BIM based software. By increased utilization – and demand from the of AEC industry against the software developer, the lack if' I' in the 'BIM' will decrease. (Hjelseth, 2009a).

The demand for information in the IFC file will in some degree depend on what the specific rules request. We recognize that the IFC files typically contain a Coordination View (MVD) with information about project, site, building, stores, spaces, walls and slabs, beams, columns, footings, windows, doors and openings, mechanical, electrical and plumbing parts, and their relationships. Most of this is commonly produced in IFC compliant software already. In addition to this we recommend that the IFC file contain information about zone and system group definitions, ceiling elements, plenum spaces, elements and openings, type and layers of materials. An extended information set should contain some form of classification (IFC is a non-hierarchical schema) by use of

Omniclass and other regional systems. Information about the building should contain part building definitions, (apartment) and aggregate building definitions (campus). Information about shape of terrain and relationship to buildings should be included.

If the model does not contain sufficient information to evaluate a specific metric, then that metric will be deemed neither True nor False, but Unknown. The logical engine can identify the impact of the Unknown which may or may not, depending on the logical structure of the regulation, impact the status of the higher level objectives, right through to the regulation as a whole. Even if there is no BIM available, the logical statement can be used to control structured dialogues, by telephone or by web page with the user. For example, the ICC has demonstrated user interfaces and storefronts that will allow users to describe their building incrementally and apply automatic checking of defined codes and standards (Wix, Nisbet and Liebich, 2008).

11. USING THE IFC CONSTRAINT MODEL

Both simple and complex constraints can be captured using the IFC constraint sub-schema. This is done through the use of a constraint aggregation "IfcRelAggregatesConstraints" where the aggregation can be characterized by a logical AND, logical OR, logical NOTAND or logical NOTOR operators. This relationship is illustrated in figure 4.

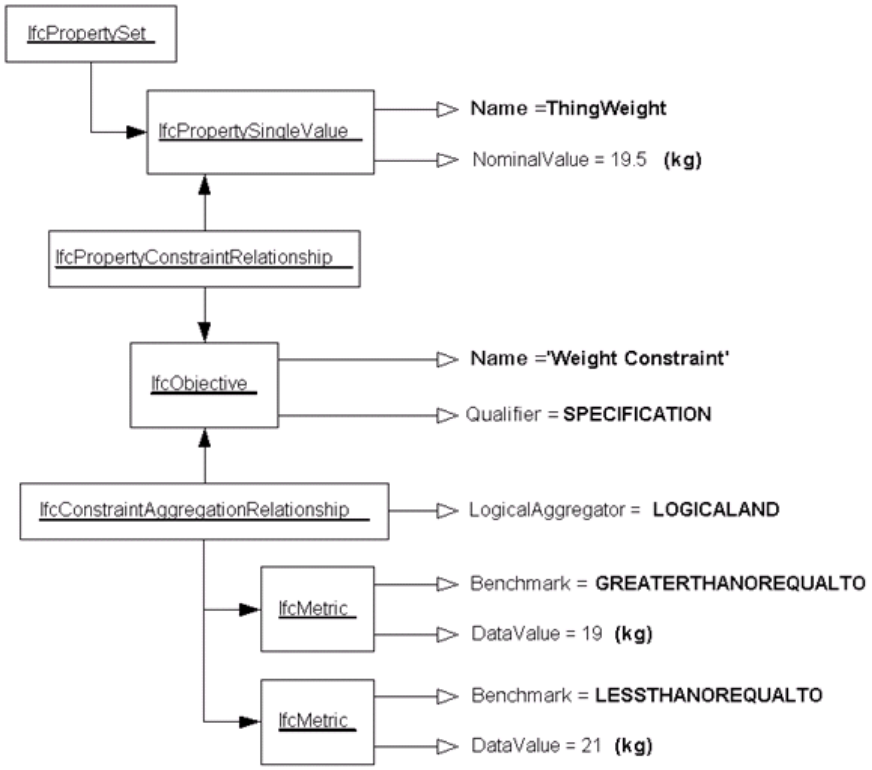

Figure 4: IFC Constraint Model

From the IFC2x3 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.

12. STANDARDS

The methodology has been described without many references to the underlying standards that make it practical and economic. In summary, W3C standards, and in particular XHTML, XML, XSLT and XLINK, allow the creation, presentation and interpretation of marked-up regulations. The general web infrastructure allows owners of documents to make them accessible dynamically in such a format without loss of control. ISO/PAS 16739:2005 "IFC2x Platform" allows all the major BIM applications to prepare descriptions of buildings in a form susceptible to automated checking based on IFCs. This means that the tools to actually generate a BIM that can be checked are already in daily use within the industry. It also provides a schema for the representation of the logical content of each regulatory framework, for use if the source regulations are not accessible. Lastly, ISO 12006-3:2007 "Framework for object-oriented information" provides an International Framework for Dictionaries (IFD) to manage the definition and sharing of common construction concepts independent of language and culture.

13. DISCUSSIONS

This paper attempt to verify two hypotheses. The first hypothesis states that use of four semantic mark-up operators; Requirement, Applies, Select, and Exception are sufficient for development of applicable rules from text such as regulations or standards. The evidence for this is based on chain of thought from philosophical and logical concepts from Aristotle to Boolean and Pierce by illustrating the strong similarities. Practical results from projects as ICC SmartCode have been successful but cannot provide absolute verification. It is accepted that the "generate and test" approach for verification does itself not give scientific evidence, just that all tested cases give an accepted result. Another aspect is if this semantic concept prefers a well defined structure of the text. If the text itself is very unstructured and unclear it may lead to instability in development of the rules. According to Tarski (1935, 1944) and Sowa (2000) the source of misinterpretation is in the formulation of codes and standard, and not in the proposed semantic concept. These preconditions is included in our verification, but we are sure that the closer the text of codes and standards follow the ISO regulative format, the more easily a trustworthy result for practical use is obtained.

The hypothesis about the possibility to auto generate applicable rule for software implementation is based on use of linking pre-defined software code to the test marked text by each of the four semantic operators. Based on the 1:1 relationship between mark-up and software code, the hypothesis can be regarded as verified. This is useful, but not sufficient to have a complete model checking system. The requirements for functionality in a model checking system is the foreseeable challenge in automatic collection for information from a BIM file in a defined format, e.g. IFC and its identification of objects and their properties and relationships.

14. CONCLUSONS

Semantic based model checking has potential to radically alter the cost/benefit balance for model checking tools We find it the two hypotheses proven by drawing up the direct connection they have to accepted philosophical and logic systems. Trustworthy rules can be specified from direct semantic interpretation of text by use of four semantic mark-up operators; Requirement, Applies, Select and Exception. We find it proven or at least made probable that this work can and should be done by AEC professionals, and not by programming experts. The connection between marked up text and generation of applicable software code is then automatic. This paper does not describe a complete model checking system, but verify principles as a foundation for systems for practical use.

15. FURTHER DEVELOPMENT

The semantic based system is according to Hjelseth and Nisbet (2010) able to handle a wide range of purposes as validating, guiding systems, adaptive and content based model checking. To date we have only applied the approach to fully normative documents that are expected to result in a pass or fail (validating model checking) . The authors believe that the approach can be expanded to include advisory documents where requirements, exceptions, applicability and selection are written with imprecision. This imprecision may affect the target values, the measured attributes, and the logical combinations of the metrics. However the appropriate flavors of 'fuzzy logic' should allow a single engine to process the uncertainty and give useful results.

REFERENCES

Bell, Bjørkhaug and Hjelseth, (2009) "Standardized computable rules". Pilot study of methods for development of computable rules Project at Standards Norway, http://www.standard.no/en/Sectors/Bygg-og-anlegg/Digital-byggeprosess/ISO-BIM-standards/Computable-rule-project/ (accessed: 2010-09-29).

Henderson, R: (2010) "MeaningDependsOnContext". http://www.c2.com/cgi/wiki?MeaningDependsOnContext (accessed: 2010-09-29).

Hjelseth, E. (2009a) "Exchange of relevant information in BIM-objects defined by the Life cycle Information Model (LIM) ". Paper presented at CIB-IDS, Finland, Espoo, July 10-12th 2009. http://www.optima.no/BIM/PhD_Eilif_Hjelseth/Eilif%20Hjelseth_Ref_nr_23_Paper_CIB-IDS_2009.pdf (accessed: 2010-09-29).

Hjelseth, E. (2009b) "Foundation for development of computable rules". Presented at CIB-W78, Turkey, Istanbul, Oct. 1-3th 2009. http://www.optima.no/BIM/PhD_Eilif_Hjelseth/Eilif_Hjelseth_UMB-Norway_Paper_73_CIB-W78_2009.pdf (accessed: 2010-09-29).

Hjelseth, E. and Nisbet, N. (2010). "Overview of concepts for model checking". Presented at the CIB-W078 Conference in Cairo 2010

ISO (2004) "ISO/IEC Directives, Part 2. Rules for the structure and drafting of International Standards", 5th Edition. http://www.iso.org/iso/standards_development/processes_and_procedures/drafting_standards/guidelines_for_the_preparation_and_submission_of_texts.htm (accessed: 2010-09-29).

ISO 12006-3:2007. Building construction -- Organization of information about construction works -- Part 3: Framework for object-oriented information. http://www.iso.org/iso/iso_catalogue (accessed: 2010-09-29).

ISO/PAS 16739:2005. Industry Foundation Classes, Release 2x, Platform Specification (IFC2x Platform), http://www.iso.org/iso/iso.catalogue (accessed: 2010-09-29).

Malone, B. (2002) "On the Financial Impact of MDO on the Corporation". 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimisation, Atlanta, GA.

Nisbet, N., Wix, J. and Conover, D. (2008) "The future of virtual construction and regulation checking". Chapter 17 in Virtual Futures for Design, Construction & Procurement. Edited by Peter Brandon and Tuba Kocaturk, ISBN: 978-1-4051-7024-6

Oxman R. (2006) "Theory and design in the first digital age". Design Studies 27 (3): 229-266. Special issue on Digital Design (ed. R. Oxman).

Oxman, R. (2008) "Emerging paradigms and models in digital design – performance-based architectural design". Chapter 1 in Virtual Futures for Design, Construction & Procurement. Edited by Peter Brandon and Tuba Kocaturk, ISBN: 978-1-4051-7024-6

Russell, S. and Norvig, P. (2010) "Artificial Intelligence: A Modern Approach", Prentice Hall Series in Artificial Intelligence. Englewood Cliffs, New Jersey. ISBN-13: 978-0-13-604259-4, ISBN-10: 0-13-604259-7. http://aima.cs.berkeley.edu (accessed: 2010-09-29).

Sowa, J. F. (2000) "Knowledge representation: Logical, Philosophical and Computational foundations". Thomson Learning. ISBN 0 534-94965-7

Sowa, J. F. (2006) "The Challenge of Knowledge Soup", Published at Research Trends in Science, Technology and Mathematics Education, Mumbai, http://www.jfsowa.com/pubs/challenge.pdf (accessed: 2010-09-29).

Sowa, J.F. (2007) “Fads and Fallacies about Logic“. IEEE Intel-ligent Systems, VivoMind Intelligence, Inc.
Tarski, A. (1935) “The concept of truth in formalized languages, Logic, Semantics and Metamathematics“, Oxford University Press, Oxford (1956), 1935, JH Woodger (trans.); First published as 'Der Wahrheitsbegrif in Den Formaliserten Sprachen', Studia Philosophica I.
Tarski, A. (1944) “The Semantic Conception of Truth and the Foundations of Semantics“, University of California, Berkeley. Published in Philosophy and Phenomenological Research 4 (1944). http://www.jfsowa.com/logic/tarski.htm (accessed: 2010-09-29).
Treldal, N. (2008) “Integrated Data and Process Control During BIM Design“. Master thesis, Department of Civil Engineering, Technical University of Denmark.
Wix, J.; Nisbet, N. and Liebich, T. (2008) “Using Constraints to Validate and Check Building Information Models“. eWork and eBusiness in Architecture, Engineering and Construction“, ECPPM 2008. Edited by Raimar Scherer and Alain Zarli. Taylor & Francis 2008. Print ISBN: 978-0-415-48245-5. eBook ISBN: 978-0-203-88332-7.