A FRAMEWORK FOR AUTOMATING CODES CONFORMANCE IN STRUCTURAL DOMAIN

Nawari. O. Nawari, Ph.D., P.E.

ABSTRACT

The concept of intelligent codes (SMARTcodes) is a new initiative of the International Code Council (ICC) in coordination with buildingSmart Alliance that strives to automate code compliance check which takes the building plan as represented by a Building Information Model (BIM), and instantly checks for code compliance via model checking software. The goal is to be able to create an inspection checklist of building elements to look for, and viewing the building components that don't comply with code provisions and for what reasons.

This paper examines automated code compliance checking systems that assess building designs according to various structural code provisions. This includes evaluating and reviewing the functional capabilities of both the technology and schema of smart codes and current building design rule checking systems. The paper proposes a new framework for development of automated rule checking systems to verify structural design against code provisions and other user defined rules.

INTRODUCTION

Currently structural design and construction processes become more complex every day because of the introduction of new building technologies, research outcomes and increasingly stringent building codes. As a result, structural engineers are responsible to comply with many regulations and specifications ranging from seismic, blast resistance, progressive collapse, to fire safety and energy performance requirements. They are constantly facing the problem of checking the conformance of products and processes to international, national and local regulations. They are also more and more subject to increasing expectations on several knowledge domains, striving towards building designs with better performance and quality. These challenges require an intense collaboration among project participants, and a profound verification of the building design starting from the earliest stages in the design process.

The introduction of Smart Codes will greatly improve the current design practice by simplifying the access to code provisions and complaints checks. Converting Code and Standards from a textual rigid format into digitally dynamic actionable format does play the key role. By breaking through the precincts of Code and Standard provisions, design software, and the Building Information Modeling a solution to insurmountable hurdle can be achieved.

Smart or intelligent code is referred to as the electronic digital format of the building codes that allow automated rule and regulation checking without modifying a building design, but rather assesses a design on the basis of the configuration of parametric objects, their relations or attributes. Smart Codes employ rule-based systems to a proposed design, and give results in format such as “PASS”, “FAIL” or “WARNING”, or ‘UNKNOWN’’ for conditions where the required information is incomplete or missing.

There has been a long historical interest in transforming building codes into a format acquiescent for machine interpretation and application. The initial effort was started in 1966 when Fenves made the observation that decision tables, an if-then-novel programming and program documentation technique, could be used to represent design standard provisions in a precise and unambiguous form. The concept was put to use when the 1969 **AISC Specification** (AISC 1969) was represented as a set of interrelated decision tables. The stated purpose of the decision table formulation was to provide an explicit representation of the **AISC Specification,** which could then be reviewed and verified by the AISC specification committee and subsequently used as a basis for preparing computer programs. Subsequently, Lopez et al. implemented the SICAD (Standards Interface for Computer Aided Design) system (Lopez and Elam 1984; Lopez and Wright 1985; Elam and Lopez 1988; Lopez et al. 1989). The SICAD system was a software prototype developed to demonstrate the checking of designed components as described in application program databases for conformance with design standards. The SICAD concepts are in production use in the AASHTO Bridge Design System (AASHTO 1998). Garrett developed the Standards Processing Expert (SPEX) system (Garrett and Fenves 1987) using a standard-independent approach for sizing and proportioning structural member cross-sections. The system reasoned with the model of a design standard, represented using SICAD system representation, to generate a set of constraints on a set of basic data items that represent the attributes of a design to be determined.

Then further research effort was led by Singapore building officials, who started considering code checking on 2D drawings in 1995. In its next development, it switched and started the CORENET System working with IFC (Industry Foundation Classes) building models in 1998 (Khemlani,, 2005). In the United States similar works have been initiated under the Smart Code initiative. There are also other several research implementations of automated rule-checking to assess accessibility for special populations (SMC, 2009) and for fire codes (Delis, 1995). The GSA and US Courts has recently supported development of design rules checking of federal courthouses, which is an early example of rule checking applied for automating design guides (GSA, 2007) .

More focused research efforts on frameworks for the representation and processing of design standards for automated code conformance began two decades ago (Yabuki and Law 1992; Kiliccote 1996). During that time, building models and the methods for rule checking have been developed, but effective Smart Codes systems are just beginning to emerge. In the 1990s, the introduction of the Industry Foundation Classes (IFC) led to early research for using this building model schema for building code checking. Han and others laid out schema for a client–server approach (Han et.al, 1997 and Vassileva,, 2000). They later developed a simulation approach of American Disability Act (ADA) wheelchair accessibility checking (Han et. al, 1999, 2002). These efforts set the stage for larger, more industrial-based efforts. A comprehensive survey of developments for computer representation of design codes and rule checking was reported by Fenves et al. (1995) and Eastman et al. (2009).

SMART CODES

This refers to the electronic digital representation of the rules and regulations of the building codes and the dictionary needed for that format. In the United State, the International Codes Council (ICC) will be available in a some form of XML. To maintain consistency of properties within the digital format of the Codes a dictionary of the properties found within the building codes is being developed (Figure 1). The dictionary is being developed as part of the International Framework for Dictionaries (IFD) effort and, in the US, is being managed by the Construction Specifications Institute (CSI) in cooperation with ICC. This work is also enabling the properties within the codes to be identified against appropriate tables within the Omniclass classification system that has been developed by CSI.

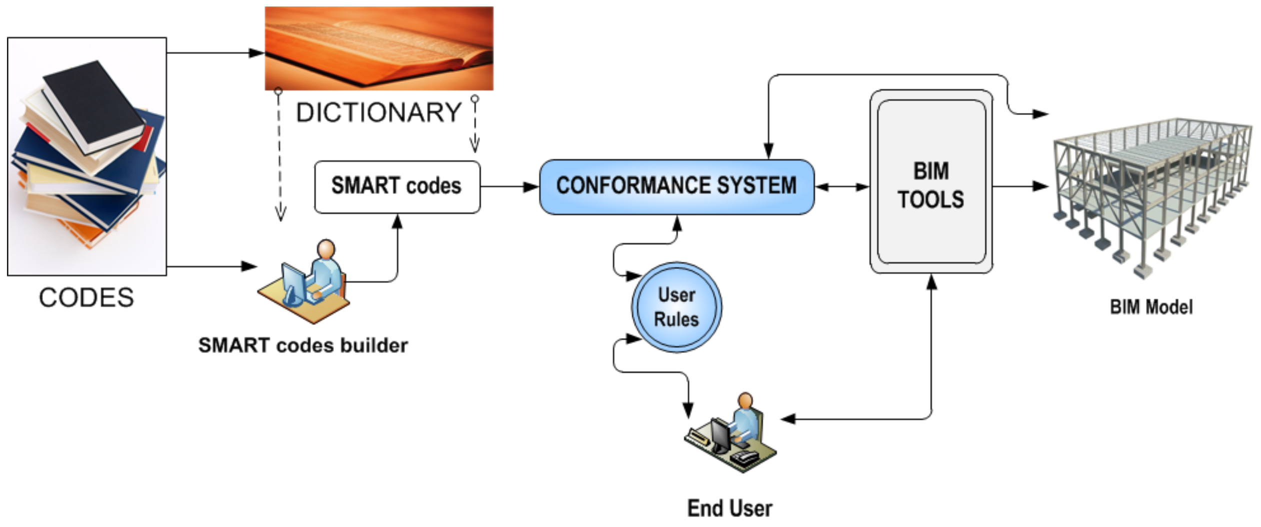

Figure 1. Automated Code Checking.

Recently, a number of researchers investigated the application of ontology-based approach (Yurchyshyna et al. 2009) and the semantic web information as a possible rule checking framework (Pauwels et. al. 2009). The first research approach works on formalizing conformance requirements conducted under the following methods (Yurchyshyna et al. 2009): (i) knowledge extraction from the texts of conformance requirements into formal languages (e.g. XML, RDF); (ii) formalization of conformance requirements by capitalizing the domain knowledge. (ii) semantic mapping of regulations to industry specific ontologies; and (iv) formalization of conformance requirements in the context of the compliance checking problem. On the other hand the semantic web approach focuses on enhancing the IFC model by using description language based on a logic theory such as the one found in semantic web domain Pauwels et. al. 2009). Because the IFC schema was not explicitly designed for interaction with rule checking environments, its specification is not based on a logic theory. By enhancing IFC onto a logical level, it could be possible to enable design and implementation of significantly improved rule checking systems.

As can be seen, Smart Codes systems depend on Information availability and rule conformance checking system. Each of these components has some limitations aspects. Major cluster of difficulties are related to the nature of Codes and Standards. Building Codes can be extremely subjective in certain provisions. That means legal scholars have the ability to argue either side of a question using accepted methods-of legal discourse. The most recurring cause of indeterminacy of Code provisions is caused by open-textured concepts used in expressing the provisions. An open-textured concept is one in which application to factual situations cannot be automatic, but which requires subjective decision and is context dependent.

It is clear that a powerful semantic-oriented representation that encompasses most of the Codes and Standard provisions and the encoding of the knowledge domain are keys in the success of Smart Code initiative. The paper proposes a new framework based upon XML and LINQ (Language Integrated Query) language to enable basic and complex level of rules and reasoning to be expressed both in XML as a normative concrete syntax and in a more human-readable abstract syntax to allow for effective AC3 systems.

THE ROLE OF BUILDING INFORMATION MODELING (BIM)

The primary requirement in application of Smart Codes is that object-based building models (BIM) must have the necessary information to allow for complete code checking. BIM objects being created normally have a family, type and properties. For example, an object that represents a structural columns possess type and properties such as steel, wood or concrete, and sizes etc. Thus the requirements of a building model adequate for code conformance checking are stricter than normal drafting requirements. Architects and Engineers creating building models that will be used for code conformance checking must prepare them so that the models provide the information needed in well-defined agreed upon structures. The BIM models created by typical BIM platform such as REVIT and ARCHICAD to date do not typically include the level of detail needed for building code or other types of rule checking. The GSA BIM Guides (GSA, 2009) provide initial examples of modeling requirements for simple rule checking. This information must then be properly encoded in IFC by the software developers to allow proper translation and testing of the design program or the rule checking software. IFC is currently considered one of the most appropriate schemas for improving information exchange and interoperability in the construction industry. New applications have been developed, capable of parsing IFC models, interpreting and reusing the available information. These software applications have mainly concentrated on deriving additional information concerning specialized domains of interest. The code conformance domain represents a new level of details and requirements on IFC model. This should be achieved by developing the appropriate Information Delivery Manuals (IDMs) and Model View Definitions (MVDs) for the Automated Code Conformance Checking (AC3) domain. For instance figure 2 below depicts the process map of the structural design IDM (Nawari 2010) while figure 3 expands the illustration of the exchange requirements for code conformance checking of the design review tasks.

Development of the required model views goes hand-in-hand with the preparation of code conformance checking functions. Code conformance checking can be constructed upon different types of model views in response to the exchange requirements specified in the IDM. An example of these model view for code checking some sections of codes has been developed by International Code Council ( ). This Model View is intended to enable BIM based automated building code compliance checking. The scope for this version includes code provisions from the International Energy Conservation Code (IECC, 2006).

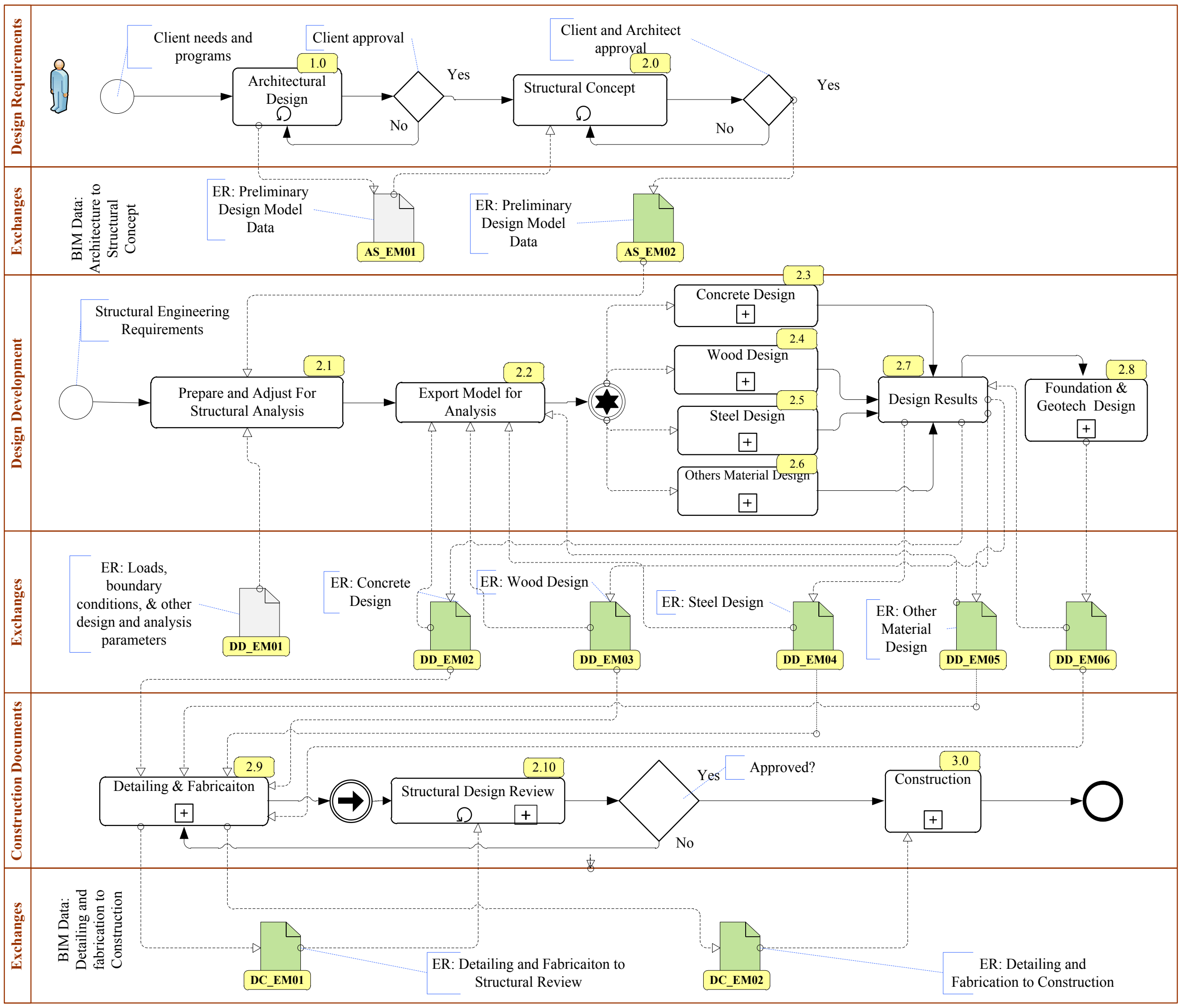
Design Requirements
Client needs and programs
1.0
Architectural Design
Client approval
Yes
No
Client and Architect approval
2.0
Structural Concept
Yes
No
Exchanges
BIM Data: Architecture to Structural Concept
ER: Preliminary Design Model Data
AS_EM01
ER: Preliminary Design Model Data
AS_EM02
Design Development
Structural Engineering Requirements
2.1
Prepare and Adjust For Structural Analysis
2.2
Export Model for Analysis
2.3
Concrete Design
2.4
Wood Design
2.5
Steel Design
2.6
Others Material Design
2.7
Design Results
2.8
Foundation & Geotech Design
Exchanges
ER: Loads, boundary conditions, & other design and analysis parameters
DD_EM01
ER: Concrete Design
DD_EM02
ER: Wood Design
DD_EM03
ER: Steel Design
DD_EM04
ER: Other Material Design
DD_EM05
DD_EM06
Construction Documents
2.9
Detailing & Fabricaiton
2.10
Structural Design Review
Yes
Approved?
No
3.0
Construction
Exchanges
BIM Data: Detailing and fabrication to Construction
DC_EM01
ER: Detailing and Fabricaiton to Structural Review
DC_EM02
ER: Detailing and Fabrication to Construction

Figure 2. Process Map of the Structural IDM.

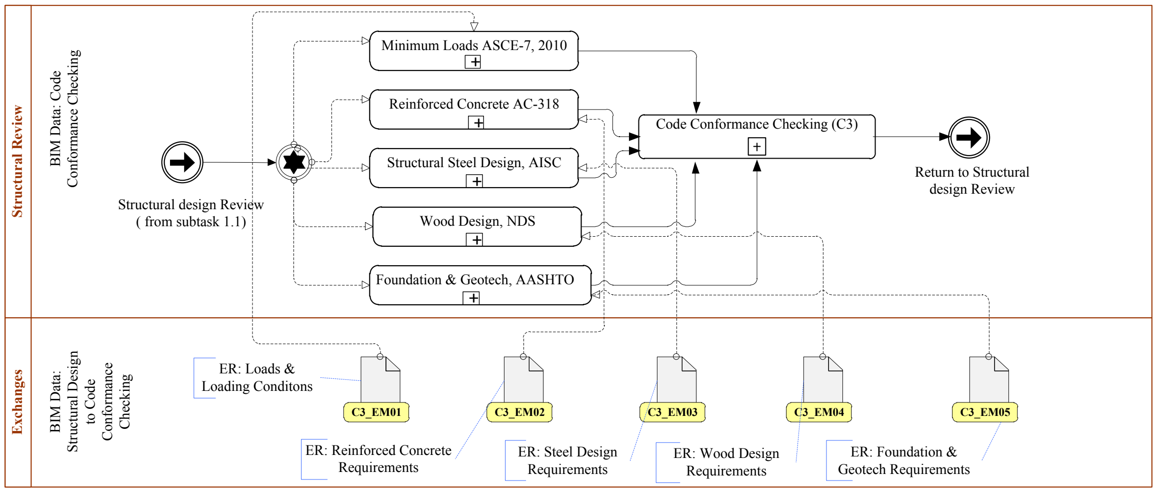
Structural Review
BIM Data: Code Conformance Checking
Structural design Review (from subtask 1.1)
Minimum Loads ASCE-7, 2010
Reinforced Concrete AC-318
Structural Steel Design, AISC
Wood Design, NDS
Foundation & Geotech, AASHTO
Code Conformance Checking (C3)
Return to Structural design Review
Exchanges
BIM Data: Structural Design to Code Conformance Checking
ER: Loads & Loading Conditons
C3_EM01
C3_EM02
C3_EM03
C3_EM04
C3_EM05
ER: Reinforced Concrete Requirements
ER: Steel Design Requirements
ER: Wood Design Requirements
ER: Foundation & Geotech Requirements

Figure 3. Process Map Illustrating the Exchange Requirements for the AC3 Framework.

AUTOMATED RULE-BASED EVALUATION SYSTEMS

Currently there are a number of different software platforms that have been developed to support implementation aspects of code provision checking systems. They vary generally in their capability of automating design checking process, flexibility of modeling design information, flexibility of encoding building codes and domain knowledge, capability of providing friendly reporting systems and 3D visualization, and the ability of integrating with other applications. Three of these commonly used platforms are briefly described herein:

Solibri Model Checker (SMC):

SMC is a JAVA-based stand along platform application that reads an IFC model and maps it to an internal structure facilitating access and processing (SMC, 2009). It includes a variety of built-in functions: such as a library for pre-checking a model, such as shape overlaps, name and attribute conventions, object existence, Fire code exit, path distance checking, space program checking against the actual spaces in a building and others. It also offers automatic viewing of checking issues along with a variety of means for reporting checking issues that include pdf, xml, and xls formats, as well as proprietary SMC visualization and reporting format suitable for design reviews using the free Solibri Model Viewer. Rules can be parametrically varied through table-set control parameters. However, entirely new rules can be added in JAVA using the SMC application programming interface (API). The API interface is not publicly available, but can be requested from Solibri.

Jotne EDModelChecker (EDM):

EDM provides an object database and supports the open development of rule checking using the EXPRESS language, which is the language in which the IFC model schema is written. New model views can be developed using EXPRESS and EXPRESS-X, which is a language for mapping instance data from one EXPRESS schema to another and supports extensive queries and reports (ISO, 1997, 1999). These facilities make EDM open to sophisticated user extensions. EDMalso provides textual reporting and server services. It is supported by EDMModel Server, an object-based backend database server, that allows EDM to deal with large building models and potentially several of them at a time (EDM, 2009).

FORNAX:

FORNAX is the first substantial effort in building code checking represented by the Singapore

CORENET when created its own platform, called FORNAX, developed by novaCITYNETS Pte. Ltd on top of EDM Model Checker (EDM, 2009). FORNAXt is a C++ object library that derives new data and generates extended views of IFC data. FORNAX objects carry rules for assessing themselves, providing good object-based modularity. It has been reviewed by a number of other building code efforts as a possible platform including the Norwegian Selvaag Group, who experimented with it to check fire exit provisions(Selvaag, 2007).

Proposed Automated Code Conformance Checking Framework (AC3)

The suggested rule-based checking system is based upon a number enabling technologies described earlier. Namely, these are XML smart Codes, BIM, and LINQ (Language Integrate Query) LINQ. The framework schema of this platform is depicted in figure 4 below:

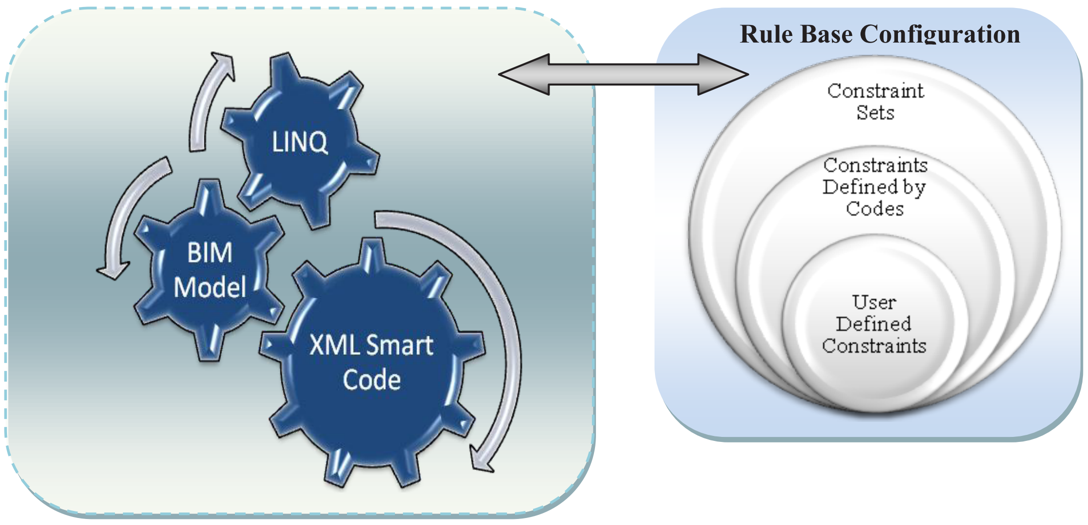

Figure 4: Automated Code Conformance Checking Framework (A**C3**)

In this framework the BIM model data is represented in ifcXML and FBM (Feature-based Model) as suggested by Nepal et. al. 2008. Due to the complicated query paths and sometimes requirements of multiple separated queries or functions, extracting features/properties from the original ifcXML is quite complicated leading to performance degradation. Thus, the FBM is introduced to improve performance and simplicity. It is an intermediate schema in XML to store information that extracted from ifcXML to enable code conformance checking. It is sometimes referred to as FBM-xml. The schema of FBM-xml is really simple: every instance of a feature is an element; all properties of a feature with their values are explicitly represented as sub-elements. The FBM-xml system instantiates feature instances and property values by directly extracting explicitly-defined components and by analyzing the geometry and topological relationships between objects in the IFC model to derive implicitly-defined features. The result is an XML data model tailored for AC3 in structural design domain (see figure 5).

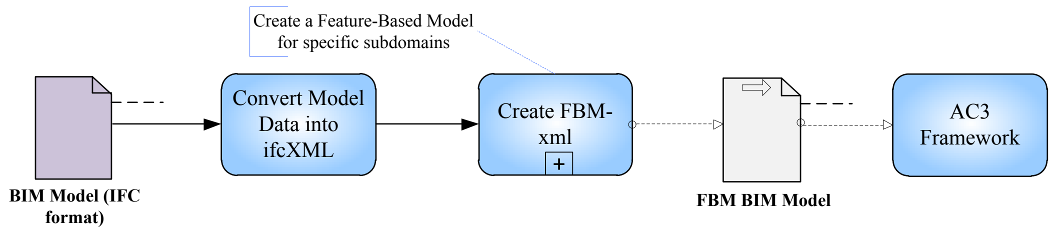
Create a Feature-Based Model for specific subdomains
Convert Model Data into ifcXML
Create FBM-xml
AC3 Framework
BIM Model (IFC format)
FBM BIM Model

Figure 5. Preparing BIM model for AC3 Processing

This study focuses on developing a framework for rule-based checking systems utilizing LINQ and XML Smart Codes. The LINQ (Language-integrated query) technology as a part of Microsoft .Net framework allows *query expressions* to benefit from the rich metadata, compile-time syntax checking, static typing and IntelliSense. Language-integrated query also allows a single general purpose declarative query facility to be applied to all in-memory information, not just information from external sources. The .NET Language-Integrated Query defines a set of general purpose *standard query operators* that allow traversal, filter, and projection operations to be expressed in a direct yet declarative way in any programming language. The standard query operators allow queries to be applied to any **IEnumerable<T>**-based information source. LINQ allows third parties to augment the set of standard query operators with new domain-specific operators that are appropriate for the target domain or technology. More importantly, third parties are also free to replace the standard query operators with their own implementations that provide additional services such as remote evaluation, query translation, and optimization. By adhering to the conventions of the LINQ pattern, such implementations enjoy the same language integration and tool support as the standard query operators.

More specifically, the framework suggested focused on LINQ to XML based data. It is in essence a LINQ-enabled, in-memory XML programming interface that facilitates communicating with XML from within the .NET Framework programming languages. The powerful extensibility of the query architecture used in the LINQ provides implementations that work over both XML and SQL data stores. The query operators over XML (LINQ to XML) use an efficient, easy-to-use, in-memory XML facility to provide XPath/XQuery functionality in the host programming language.

To illustrate the concept of the AC3 system, XML file is created for a part of the ACI 318-05 Code (ACI 318, 2005) and depicted in Figure 6 below.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ACI318>
  <Year year="2005">
    <Section Number = "7.7" title="Concrete Protection for Reinforcement">
      <SubSection Number="7.7.1" title="Cast-in-place concrete (nonprestressed)">
        <Category title ="Concrete cast against and permanently exposed to earth" >
          <MinimumCover> 3 </MinimumCover>
        </Category >
        <Category title ="Concrete exposed to earth or weather" >
          <Rebar Min="#6" Max="#18" Members="All">
            <MinimumCover> 2 </MinimumCover>
          </Rebar>
          <BarSizes Min="#3" Max="#5" Members="All">
            <MinimumCover> 1.5 </MinimumCover>
          </BarSizes>
        </Category >
        <Category title ="Concrete not exposed to weather or in contact with ground" >
          <Rebar Min="#14" Max="#18" Members="Slabs, Walls, Joists">
            <MinimumCover> 1.5 </MinimumCover>
          </Rebar>
          <Rebar Min="#3" Max="#11" Members="Slabs, Walls, Joists">
            <MinimumCover> 0.75 </MinimumCover>
          </Rebar>
          <Rebar Min="#3" Max="#18" Members="Beams, Columns">
            <MinimumCover> 1.5 </MinimumCover>
          </Rebar>
          <Rebar Min="#6" Max="#18" Members="Shells, folded plate members">
            <MinimumCover> 0.75 </MinimumCover>
          </Rebar>
          <Rebar Min="#3" Max="#5" Members="Shells, folded plate members">
            <MinimumCover> 0.5 </MinimumCover>
          </Rebar>
         </Category >
      </SubSection>
      <SubSection Number="7.7.2" title="Cast-in-place concrete (prestressed)">
        <Category title ="Concrete cast against and permanently exposed to earth" >
          <MinimumCover> 3 </MinimumCover>
        </Category >
        <Category title ="Concrete exposed to earth or weather" >
          <Rebar Min="#3" Max="#18" Members="Wall panels, slabs, joists">
            <MinimumCover> 1 </MinimumCover>
          </Rebar>
          <Rebar Min="#3" Max="#18" Members="Beams, Columns">
            <MinimumCover> 1.5 </MinimumCover>
          </Rebar>
        </Category >
                <Category title ="Concrete not exposed to weather or in contact with ground" >
          <Rebar Min="#3" Max="#18" Members="Slabs, Walls, Joists">
            <MinimumCover> 0.75 </MinimumCover>
          </Rebar>
          <Rebar Min="#3" Max="#11" Members="Slabs, Walls, Joists">
            <MinimumCover> 1 </MinimumCover>
          </Rebar>
          <Rebar Min="#5" Max="#18" Members="Beams, Columns">
            <MinimumCover> 1.5 </MinimumCover>
          </Rebar>
          <Rebar Min="#3" Max="#5" Members="Beams, Columns">
            <MinimumCover> 1.0 </MinimumCover>
          </Rebar>
          <Rebar Min="#6" Max="#18" Members="Shells, folded plate members">
            <MinimumCover> 0.75 </MinimumCover>
          </Rebar>
          <Rebar Min="#3" Max="#5" Members="Shells, folded plate members">
            <MinimumCover> 0.4 </MinimumCover>
          </Rebar>
        </Category >
     </SubSection>
   </Section>
  </Year>
</ACI318>
```

Figure 6. XML Data from ACI 318-05 Code

The second step in implementing the Automated Conformance Code Checking is to establish the rules schema that allow communication with the Smart Code. This will be achieved by applying LINQ to the Smart Code example shown in figure 5. This section describes how to use Language-Integrated Query with Smart Code.

Standard query operators form a complete query language for **IEnumerable<T>**. Standard query operators show up as extension methods on any object that implements **IEnumerable<T>** and can be invoked like any other method. In addition to standard query operators are query expressions for five common query operators: **Where, Select, SelectMany, OrderBy,** and **GroupBy.** Using LINQ to extract rules from Smart Code provides the following apparent advantages:

By implementing the above described **ACCC** framework, the following checking can be executed to examine minimum concrete cover requirement for reinforced concrete beam according to the ACI 318-05. The example shown below is the case of checking the beams in a single storey reinforced building frame. The LINQ code below accesses the Smart Code and read the encoded provisions given by the ACI 318-05 (figure 7) and subsequently applies them to the type of a reinforced concrete beam in the building.

```
XElement ACCC = XElement.Load("C:\Pap\BIM\SmartCode\XMLFile1.xml");
var c = ACCC...<Section>;
IEnumerable<XElement> QUERY =
     From i In c.<SubSection> Where (string) i.@title = "Concrete Protection for Reinforcement"
Select i;
ForEach (XElement i In QUERY)   {
   string pt1 = i.<Points>.Distinct.<Grade1>.Value.ToString();
   string pt2 = i.<Points>.Distinct.<Grade2>.Value.ToString();
   string pt3 = i.<Points>.Distinct.<Grade3>.Value.ToString();
}
System.Xml.XmlDocument doc = new System.Xml.XmlDocument();
doc.Load("C:\Pap\BIM\SmartCode\XMLFile2.xml");
System.Xml.XmlNodelList list = doc.GetElementsByTagName("ShellSurface");
ForEach(System.Xml.XmlElement j In list ) {
    string wType = j.GetAttribute("surfaceType");
                                       If wType == "Wall" {
      string buildinginsulation=j.Item("Insulation").InnerText; }
}
Switch (Buildinginsulation) {
      Case "grade1":
         int points = pt1;
         Break;
      Case "grade2":
         int points = pt2;
         Break;
      Case "grade3":
         int points = pt3;
         Break;
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<fbmModel>
  <feature type="Component" id="0" ifcid="ID428" ifcTitle="ifcBeam">
    <feature_type>Beam</feature_type>
    <contained_in_the_Storey> Level 1</contained_in_the_Storey>
    <material>Concrete - Cast-in-Place Concrete</material>
    <beam_type> Concrete-Rectangular Beam:12 x 24:12 x 24:24795</beam_type>
    <depth>24</depth>
    <width>12</width>
    <span>24:24795</span>
    <is_external>false</is_external>
    <fire_rating>1 hr</fire_rating>
    <is_curved>false</is_curved>
    <is_clipped>false</is_clipped>
    <has_opening>false </has_opening>
    <is_loadbearing>true</is_loadbearing>
    <has_Rebar>true</has_Rebar>
    <rebar_shape></rebar_shape>
    <rebar_bottom_size>9 </rebar_bottom_size>
    <rebar_bottom_number>3</rebar_bottom_number>
    <rebar_bottom_cover>1.5</rebar_bottom_cover>
    <rebar_top_size>9</rebar_top_size>
    <rebar_top_number>2</rebar_top_number>
    <rebar_top_cover>1.5</rebar_top_cover>
    ...
  </feature>
   …

</fbmModel>
```

Figure 8: A portion of an fbmXML produced from a BIM model for a building.

DISCUSSION

The proposed AC3 framework provides an object model that is lighter weight and easier to work with, and that takes advantage of modern programming languages. The most important advantage of the AC3 framework lies in the integration power of Language-Integrated Query (LINQ). This integration enables encoding queries on the in-memory XML document to retrieve collections of elements and attributes. The integration of LINQ in modern programming platform provides stronger typing, compile-time checking, and improved debugger support. Further benefits of this framework include the ability to use query results as parameters to XElement and XAttribute object constructors enables a powerful approach to creating XML trees. This approach, called functional construction, facilitate the easy transformation of XML trees from one shape to another.

The current evaluation of Smart Codes and the automated model checking technologies to assist building analysis and design practices demonstrated an immense advancement to achieve an optimum design. Examples include more complete and accurate performance estimates earlier in the design process, improved life-cycle costing analysis, increased opportunities for measurement and verification during building occupation, and improved processes for gathering lessons learned in high performance building. In general, advancements in these technologies will increase the role Smart Codes and model checking play during both design and building operation, leading to an overall optimization in building design.

Notwithstanding the significant advantage that the intelligent cods technologies is promising, some difficulties still persist. For instance, the automatic verification demands the interpretations of building codes information and performance requirements supported by the domain knowledge, which is basically the regulations and provisions that are first defined by people and represented in human language formats, typically written text, tables and equations. In building codes, these provisions have legal status. How can the interpretation of these rules into a digital format be done, without violating the written rules? Currently, in some Code conformance checking implementations, the process relies on the software developer's interpretation and translation of the written rules into computer-based conformance evaluation. In other cases, the logic of the human language statements in the Codes is formally interpreted and then translated into a machine executable format. Furthermore, some important design rules apply to properties that require complex simulations or analyses, such as for structural integrity or energy usage. These require the application of an analysis model to derive the complex information, then to apply the rules to the analytically derived data. Other issues deal with missing information of the model view of the building. Having the code conformance checking system derive new data or generate model views that explicitly derive the lacking data represent vulnerability and legal risks.

CONCLUSIONS

Application of the AC3 framework in structural design has the impending to optimize and simplify the automated code and standard conformance checks by leveraging building information that exists in the architectural and structural models created by BIM authoring platform. The proposed automated code conformance checking (AC3) framework has many advantages over existing rule checking systems. The major differentiator of the AC3 lies in the abilities of LINQ to XML as in-memory XML programming platform. Language-Integrated Query provides a consistent query experience across different data models as well as the ability to mix and match data models within a single query, it is able to depict an unlimited range of rules, including unlimited nested conditions and branching of alternative contexts within a specified domain. Furthermore, AC3 provides flexibility of encoding building codes provisions and domain knowledge, capability of providing friendly user-defined rules, and the ability of integrating with other applications. Increasing BIM adoption and the concomitant increasing interest in the interoperability potential of XML prove to be the essential catalyst in the successful adoption and further advancement of automated code conformance checking (AC3) systems.

ACKNOWLEDGEMENT

The author would like to express his appreciation to College of Design, Construction & Planning, University of Florida, Gainesville, Florida for funding and supporting this research.

REFERENCES

Conover, D. (2007).”*Development and Implementation of Automated Code Compliance Checking in the U.S.*”, International Code Council, 2007.

Delis, E.A., and Delis, A. (1995). “*Automatic fire-code checking using expert-system technology*”, Journal of Computing in Civil Engineering, ASCE 9 (2), pp. 141–156.

Ding, L., Drogemuller, R., Rosenman, M., Marchant, Gero, D. J. (2006). “ *Automating code checking for building designs: in: K. Brown, K. Hampson, P. Brandon (Eds.), Clients Driving Construction Innovation*”: Moving Ideas into Practice, CRC for Construction, Innovation, Brisbane, Australia, pp. 113–126.

Eastman, C. M., Jae-min Lee, Yeon-suk Jeong, Jin-kook Lee (2009). “*Review Automatic rule-based checking of building designs* “, Journal of Automation in Construction (18), pp. 1011–1033, Elsvier.

EDM (2009).” EXPRESS Data Manager”, EPM Technology, http://www.epmtech.jotne.com.

Fenves, S. J. (1966). “ *Tabular decision logic for structural design*”, *J.* Structural Engn *9* 92, pp. 473-490

Fenves, S. J. and Garett Jr, J. H. (1986). “*Knowledge-based standards processing”,* Int. J. Artificial Intelligence Engn 1, pp. 3-13.

Fenves, S. J., Garrett, J. H., Kiliccote. H., Law. K. H., and Reed, K. A. (1995). "*Computer representations of design standards and building codes: U.S. perspective."* The Int. J. of Constr. Information Technol*.,* 3(1), pp. 13-34.

Garrett, J. H., Jr., and S. J. Fenves, (1987). “*A Knowledge-based standard processor for structural component design*” Engineering with Computers, 2(4), pp 219-238.

GSA (2007). “*U.S. Courts Design Guide*”, Administrative Office of the U.S. Courts, Space and Facilities Division, GSA, http://www.gsa.gov/Portal/gsa/ep/contentView.do?P=PME&contentId=15102&contentType=GSA_DOCUMENT .

GSA (2009). “*BIM Guide for Circulation and Security Validation*”, GSA Series 06 (draft).

Hietanen, J. (2006). “IFC Model View Definition Format”, International Alliance for Interoperability.

ICC (2006). “*MDV for the International Energy Conservation Code*”, http://www.blis-project.org/IAI-MVD/.

ISO TC184/SC4 (1997). “ *Industrial automation systems and integration—Product data representation and exchange*” , ISO 10303-11: Description Methods: The EXPRESS Language Reference Manual, ISO Central Secretariat.

ISO TC184/SC4 (1999).” *Industrial automation systems and integration—Product data representation and exchange*:”, ISO 10303-14: Description Methods: The EXPRESS-X Language Reference Manual, ISO Central Secretariat.

Jeong, Y-S., Eastman, C.M., Sacks, R., Kaner, I. (2009) “Benchmark tests for BIM data exchanges of precast concrete”, *Automation in Construction 18 (2009) 469–484.*

Khemlani, K. (2005). “ *CORENET e-PlanCheck: Singapore's automated code checking system”*, AECBytes, http://www.aecbytes.com/buildingthefuture/2005/CORENETePlanCheck.html.

Lopez, L. A., and S. L. Elam (1984). " *SICAD: A Prototype Knowledge Based System for Conformance Checking and Design"*, Technical Report, Department of Civil Engineering. University of Illinois at Urbana-Champaign, Urbana-Champaign, IL.

Lopez, L. A., and R. N. Wright (1985). "*Mapping Principles for the Standards interface for Computer Aided Design*", NBSIR 85-3115, National Bureau of Standards, Gaithersburg, MD.

Lopez, L. A., S. Elam and K. Reed (1989). " Software concept for checking engineering designs for conformance with codes and standards". Engineering with Computers, 5, pp.63-78.

Nawari, N. O. (2009). "Intelligent Design Codes", The Structures Congress, 2009, Structural Engineering Institute, ASCE, pp.2303-2312.

Nawari, N. O. (2010)."Standardization of Structural BIM", ASCE 2011 Workshop of Computing in Civil Engineering, Florida, Miami, June 19-22, 2011.

SMC (2009). "automated code checking for accessibility" Solibri, http://www.solibri.com/press-releases/solibri-model-checker-v.4.2-accessibility.html

Vassileva, S. (2000). "An approach of constructing integrated client/server framework for operative checking of building code", in Taking the Construction Industry into the 21st Century, Reykjavik, Iceland, ISBN: 9979-9174-3-1, June 28–30 2000.