

# Code Checking and Visualization of an Architecture Design

Rong Xu<sup>1,2</sup>, Wawan Solihin<sup>2</sup>, and Zhiyong Huang<sup>1</sup>

<sup>1</sup> School of Computing, National University of Singapore

<sup>2</sup> novaCITYNETS Pte. Ltd., Singapore

Email: huangzy@comp.nus.edu.sg

## 1 Introduction

Computer graphics has been successfully applied to architecture design. There is more demand for new applications. One of them, to be addressed in this work, is the code checking and visualization of the checking results.

The architecture industries are facing more and more regulations and codes of practice [7, 8]. Each year, governments spend a huge amount of manpower into building plan approval. This process can be greatly improved if automatic checking and visualization tools are provided.

Existing work includes [1, 2, 3, 5, 6]. The major problem is that they depend on a specific data format, in particular, lacking of the semantics for a more general code checking. The results are not visualized in 3D directly.

In this poster, we report a novel solution. The major contribution is to define and use a general framework, referred to as Code Checking Object Model (CCOM). Algorithms are devised to extract the semantics information such as the relationships and the behaviors of the building elements from the CAD data in the standard IFC format [4]. The extraction results of the semantics information are represented in CCOM. For code checking, the CCOM data are used directly for each rule. The checking can be submitted via internet and the results will be visualized in the web browser. Our solution has been implemented in FORNAX, a product of NovaCITYNETS, <http://www.novacitynets.com>.

## 2 CCOM

CCOM is a hierarchical object model: An *object* is either a *container* or an *element*. A *container* contains both the *element* and *aggregation*. An *element* is the basic component consisting of the *attribute list* and *geometry* of the building element. An *aggregation* represents the semantics information using different data structures. They are selected according to the nature of the semantics information. For example, for a living room and windows, the semantics is “belongs to” and tree is most suitable.

To extract the semantics information, more than one CAD models are usually involved. For example, for the code checking, a kitchen model has distinct characteristics that are related to specific checking rules: it should have higher fire rating than other space, independent smoke control system, etc. Thus, all the related CAD models need to be traversed and checked.

The typical semantics information includes spatial, network, and design constraints. Spatial information in CCOM describes the building object location relative to others. For example, a discharge stack in the sewerage system is defined by series of connected pipes of the same size and in vertical alignment. Network information allows to find connections and access paths from one location to another. For example, a group of rooms can be networked by walking path or water pipes. Design constraints define the constraints of a group of elements. For example, a constraint may exist between the size of the space and thickness of the wall.

Geometric operations are applied in the extraction process. They are implemented in ACIS. For example, the spatial operations of the objects include contain, intersect and surround. Such information is useful when we need to determine whether a certain service system can serve a particular space. It is checked to see if the space contains at least one of the terminals. For a building model, the checking is often required against a set of spaces that is typically defined as zone. In this case, we need to derive all the spaces of the zone from the CAD models and fuse them together to form a zone before the spatial operations like contain can be applied.

The checking results are visualized. OpenGL is used in the implementation. The design components that failed to pass the code checking are highlighted in the 3D model directly.

## 3 Results

First, we use a real clause. In the *Code of Practice* from FSB, Chapter 29, clause 2.2.2, each space should be within 38 meters from landing valves. The clause is described as “The number and distribution of rising mains shall be such that all parts of the floor not more than 24m above the ground level is within 38m from landing valves.

The distance should be measured along a route suitable to hose lines, including distance any up or down stairway.” The process of code checking based on the CCOM is summarized as follows:

```

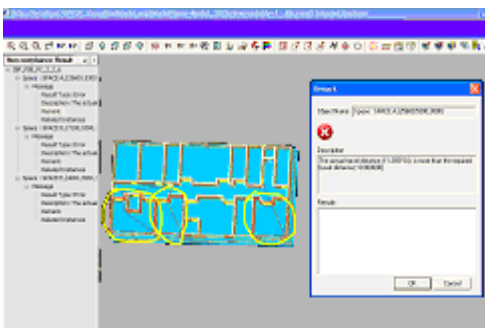
Storey = Building.getAllStorey( )
For each Storey
  Space = Storey.getAllSpace( )
  LandingValve = Storey.getAllLandingValve( )
  For each Space
    RemotePoint = Space.getRemotePoint( )
    For each LandingValve
      TravelDistance =
      Storey.computeTravelDistance(RemotePoint, Landing-
      Valve)
      If TravelDistance < 38m
        Space passed code checking
      Space code checking fail
  
```

The checking result is shown in Figure 1. The highlighted space is failed because it is too far away from a landing valve.

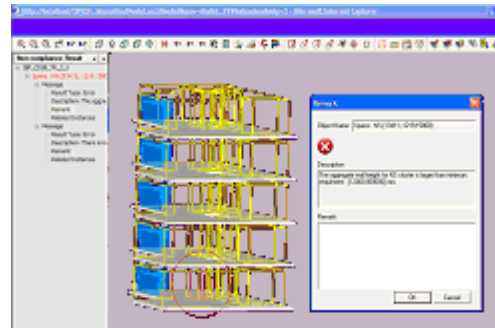


Figure 1. Visualization of the code checking result using the Code of Practice from FSB, Chapter 29, clause 2.2.2

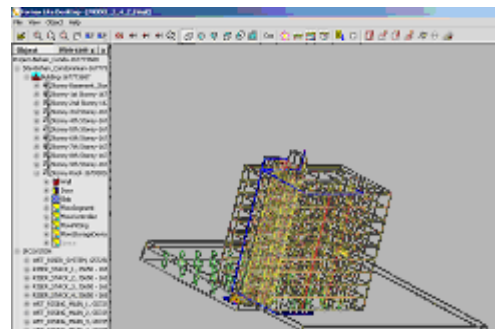
Note that in the example, only the CCOM objects *Building*, *Storey*, and *Space* can hold and access the semantics information. Four more examples of code compliance checking and visualization results are shown in Figure 2.



(a) Code compliance checking for the rule that each space should be within 10 meters from at least one exit



(b) Code compliance checking for the rule that the boundary wall height must be within 12m



(c) Code compliance checking for the rule that each apartment has at least one connection to the water tank on the roof

Figure 2. Examples of code compliance checking and visualization of the results

## References

- [1] C. S. Han, J. Kunz and K. H. Law, Making Automated Building Code Checking a Reality, *Facility Management Journal*, September/October, 1997, pp. 22-28.
- [2] C. S. Han, J. Kunz and K. H., Law, A Hybrid Prescriptive/Performance Based Approach to Automated Building Code Checking, *ASCE J. Computing in Civil Engineering*, 12(4):181-194, 1998.
- [3] C. S. Han, J. C. Kunz and K. H. Law, Compliance Analysis for Disabled Access, *Advances in Digital Government Technology, Human Factors, and Policy*. William J. McIver, Jr. and Ahmed K. Elmagarmid (eds) Boston Kluwer, 2002.
- [4] International Alliance for Interoperability Industry Foundation Classes, *Specifications Volumes 1-4*, Washington D. C., 1997.
- [5] G. Lau, K. H. Law, and G. Wiederhold, A Framework for Regulation Comparison with Application to Accessibility Codes, *Proceedings of the National Conference on Digital Government Research (dg.o2003)*, Boston, MA, May 18-21, 2003, pp. 251-254.
- [6] R. F. Woodbury, A. L. Burrow, R. M. Drogemuller, S. Datta, Code Checking by Representation Comparison, *International Journal of Design Computing*, 3(1), 2000, pp. 73-89.
- [7] California Building Code, California Building Standards Commission, 1998, <http://www.bsc.ca.gov/>
- [8] Singapore Code of Practice on Building Design, Building and Construction Authority, [http://www.bca.gov.sg/industry\\_programmes/buildable\\_design/legislation/codepractice.html](http://www.bca.gov.sg/industry_programmes/buildable_design/legislation/codepractice.html)