CONFERENCE THEME: VISUALISATION AND INFORMATION

Full Paper

AUTOMATED CODE CHECKING

Lan Ding
CMIT, Commonwealth Scientific and Industrial Research Organisation
lan.ding@csiro.au

Robin Drogemuller
CMIT, Commonwealth Scientific and Industrial Research Organisation
robin.drogemuller@csiro.au

Julie Jupp
Key Centre of Design Computing and Cognition, University of Sydney
jupp_j@arch.usyd.edu.au

Mike A Rosenman
Key Centre of Design Computing and Cognition, University of Sydney
mike@arch.usyd.edu.au

John S Gero
Key Centre of Design Computing and Cognition, University of Sydney
john@arch.usyd.edu.au

ABSTRACT

Most buildings constructed in Australia must comply with the Building Code of Australia (BCA). Checking for compliance against the BCA is a major task for both designers and building surveyors. This project carries out a prototype research using the EDM Model Checker and the SMC Model Checker for automated design checking against the Building Codes of Australia for use in professional practice.

In this project, we develop a means of encoding design requirements and domain specific knowledge for building codes and investigate the flexibility of building models to contain design information. After assessing two implementations of EDM and SMC that check compliance against deemed-to-satisfy provision of building codes relevant to access by people with disabilities, an approach to automated code checking using a shared object-oriented database is established.

This project can be applied in other potential areas – including checking a building design for non-compliance of many types of design requirements. Recommendations for future development and use in other potential areas in construction industries are discussed.

Keywords: Code Checking, Design Verification, Domain Knowledge, EDM, SMC

THE PAPER

1. INTRODUCTION

Buildings are required to meet various criteria from organisational, financial, environmental and social perspectives. An important social perspective is the equity of access to buildings by all members of the public. The requirements for access are defined in the Building Code of Australia (BCA). The BCA is a performance-based code, which allows the use of a range of solutions to requirements. Australian Standard (AS) 1428 "Design for access and mobility" is accepted as a deemed-to-satisfy solution within the BCA. This project takes AS 1428 as a focus of our application.

Express Data Manager^{TM} (EDM) and Solibri Model Checker^{TM} (SMC) are two major systems currently available that provide object-based rule engines. EDM provides a shared data repository and is compatible with the Industry Foundation Classes (IFC). SMC provides automated "design spell-checking" to a building model and is capable of directly interfacing to an object-based architectural CAD system.

This project uses EDM and SMC for the automated assessment of designs against AS1428. Two prototype systems using EDM and SMC are implemented for a comparative analysis. The fundamental issues that we deal with involve the:

- Capability of automating design checking process;
- Flexibility of modelling design information;
- Flexibility of encoding building codes and domain knowledge;
- Capability of interfacing to object-based CAD systems;
- Capability of providing friendly reporting systems and 3D visualization;
- Capability of integrating with other applications.

Section 2 and Section3 of this paper describe the functionality of EDM and SMC for encoding building codes and domain knowledge. They are followed by a demonstration of the performances of the EDM prototype and SMC prototype in Section 4 and Section 5. Section 6 presents an approach of using a shared EDM database for storing comprehensive design information and encoding domain-specific knowledge to support automated code checking and discussions on future development.

2. DOMAIN KNOWLEDGE AND EDM RULE BASE

Checking a building design for compliance with a given set of design requirements from building codes requires a process of design verification (Balachandran, Rosenman and Gero, 1991). The design verification demands the interpretations of design information and performance requirements supported by domain knowledge. This section describes the functionality of the EDM database for encoding domain knowledge and design requirements to support design verification.

The EDM database contains data models and schemas, Figure 1. The schemas include a model schema for defining data models, rule schema for validating data models and query schema for producing a specific view of a data model. This project focuses on the development of the EDM rule schema.

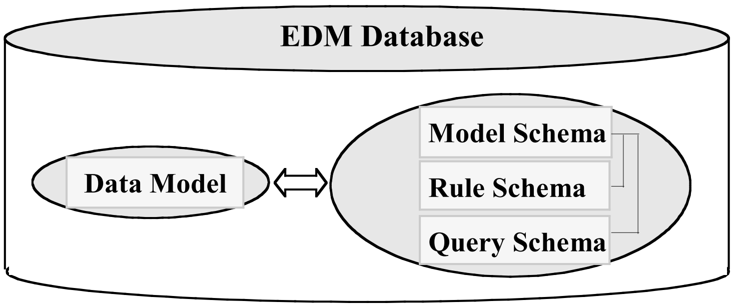


Figure 1. An illustration of the EDM database.

The EDM rule schema consists of the declarations of entities, rules, functions and procedures used for defining various rules to data models. We develop the EDM rule schema to accommodate the performance requirements and domain knowledge from building codes for design verification, Figure 2.

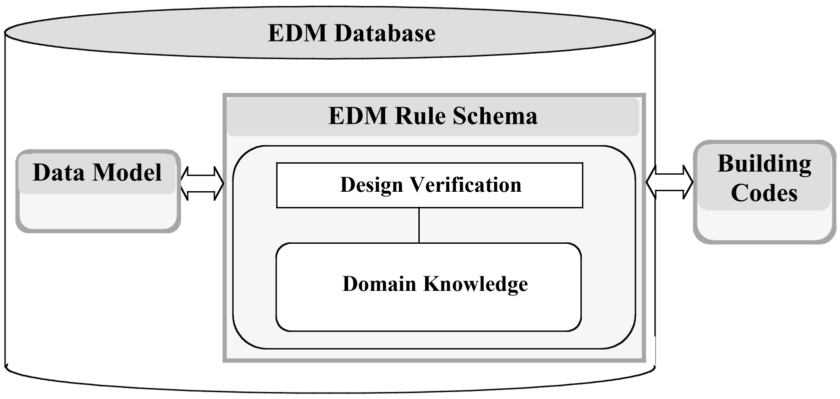


Figure 2. An illustration of using EDM rule schema for code checking.

We present an example of encoding a clause in AS 1428 Part 1 with the EDM rule schema. The clause, from 7.1 Provision of Entrances, is described as: ‘accessible entrances shall be incorporated in an accessible path of travel’. Design checking against this clause requires an interpretation on ‘accessible entrances’ and ‘an accessible path of travel’.

We develop an object-based interpretation and define it with the EDM rule schema. The object-based interpretation consists of: Descriptions, Performance Requirements, Objects, Properties, Relationships, and Domain-specific Knowledge for Interpretation. Figure 3 shows the object-based interpretation for the clause from 7.1 Provision of Entrances. A process of encoding a clause of building codes to an object-based interpretation and then to an EDM rule schema is illustrated in Figure 4.

CLAUSE 7: DOORWAYS, DOORS AND CIRCULATION SPACE AT DOORWAYS

Clause 7.1 Provision of Entrances

Description:

The requirements for entrances to buildings are as follows:
(a) Accessible entrances shall be incorporated in an accessible path of travel.

Performance Requirements:

There is an uninterrupted path of travel from an accessible entrance to an accessible space required.

Objects:

{Space, Door}

Object Properties:

{Door_exterior, Door_accessible, Door_type, Door_width, Space_accessible, Space_identification, Space_area}

Object Relationship:

{Contain (Space, Door): Space contains Door}

Domain-specific knowledge for Interpretation:
(to be implemented with functions and procedures)

AssessibleExteriorDoor (Doors)

{IF Door_exterior and Door_accessible are found, THEN return AccessibleExteriorDoors}

AccessibleEntranceSpace (AccessibleExteriorDoors)

{IF AccessibleExteriorDoors are contained by Spaces, THEN return AccessibleEntranceSpaces}

AccessibleSpaceRequired (Spaces)

{IF Space_assessible is found, THEN return AccessibleSpacesRequired}

A_Path_from_AccessibleEntranceSpace_to_AccessibleSpaceRequired (Spaces, Doors)

{IF Spaces and Doors are located in the path from AccessibleEntranceSpace to AccessibleSpaceRequired, THEN return a set of the Spaces and a set of the Doors}

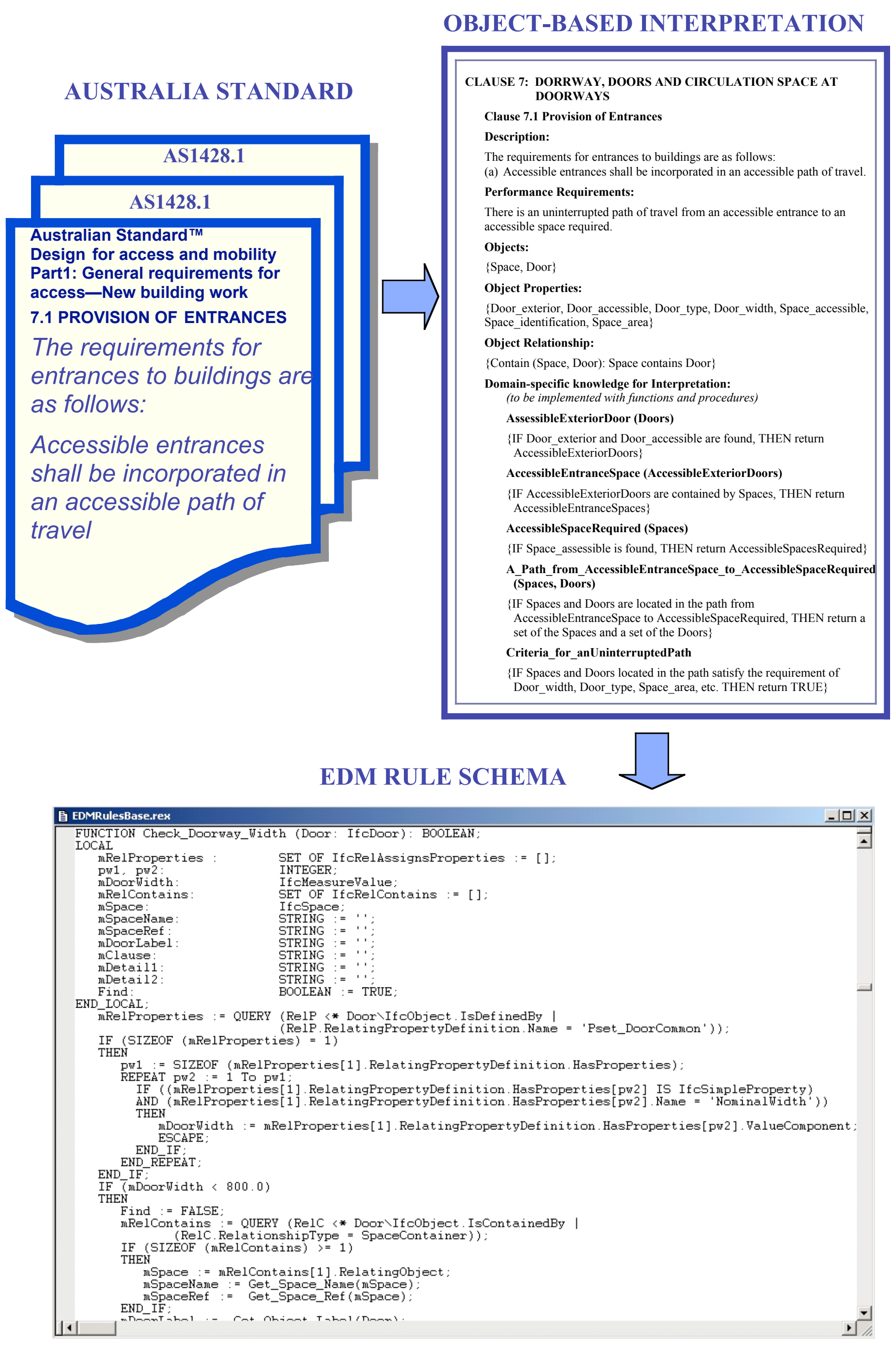
Criteria_for_anUninterruptedPath

{IF Spaces and Doors located in the path satisfy the requirement of Door_width, Door_type, Space_area, etc. THEN return TRUE}

Figure 3. An example of an object-based interpretation for a clause of building codes.

3. DOMAIN KNOWLEDGE AND SMC RULE BASE

Compared with EDM, SMC provides a Constraint Set Manager (CSM) for managing and configuring constraint sets as the rule base to support design spell-checking.



Australian Standard™
Design for access and mobility
Part1: General requirements for access—New building work

7.1 PROVISION OF ENTRANCES

The requirements for entrances to buildings are as follows:

Accessible entrances shall be incorporated in an accessible path of travel

CLAUSE 7: DORRWAY, DOORS AND CIRCULATION SPACE AT DOORWAYS

Clause 7.1 Provision of Entrances

Description:

The requirements for entrances to buildings are as follows:
(a) Accessible entrances shall be incorporated in an accessible path of travel.

Performance Requirements:

There is an uninterrupted path of travel from an accessible entrance to an accessible space required.

Objects:

{Space, Door}

Object Properties:

{Door_exterior, Door_accessible, Door_type, Door_width, Space_accessible, Space_identification, Space_area}

Object Relationship:

{Contain (Space, Door): Space contains Door}

Domain-specific knowledge for Interpretation:
(to be implemented with functions and procedures)

AssessibleExteriorDoor (Doors)

{IF Door_exterior and Door_accessible are found, THEN return AccessibleExteriorDoors}

AccessibleEntranceSpace (AccessibleExteriorDoors)

{IF AccessibleExteriorDoors are contained by Spaces, THEN return AccessibleEntranceSpaces}

AccessibleSpaceRequired (Spaces)

{IF Space_assessible is found, THEN return AccessibleSpacesRequired}

A_Path_from_AccessibleEntranceSpace_to_AccessibleSpaceRequired (Spaces, Doors)

{IF Spaces and Doors are located in the path from AccessibleEntranceSpace to AccessibleSpaceRequired, THEN return a set of the Spaces and a set of the Doors}

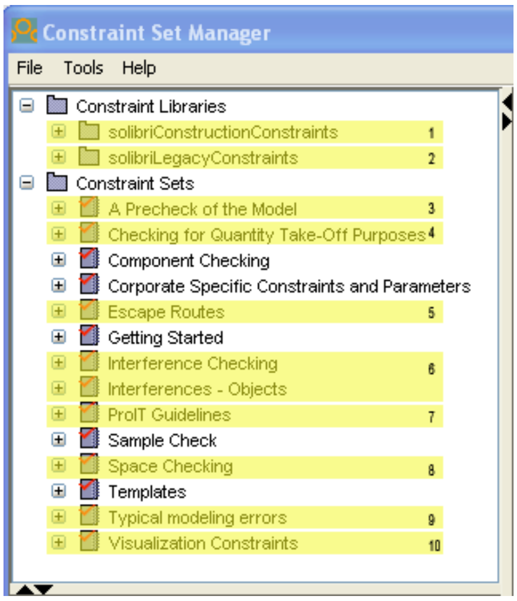
Criteria_for_anUninterruptedPath

{IF Spaces and Doors located in the path satisfy the requirement of Door_width, Door_type, Space_area, etc. THEN return TRUE}

```
FUNCTION Check_Doorway_Width (Door: IfcDoor): BOOLEAN;
LOCAL
   mRelProperties :          SET OF IfcRelAssignsProperties := [];
   pw1, pw2:                 INTEGER;
   mDoorWidth:               IfcMeasureValue;
   mRelContains:             SET OF IfcRelContains := [];
   mSpace:                   IfcSpace;
   mSpaceName:               STRING := '';
   mSpaceRef:                STRING := '';
   mDoorLabel:               STRING := '';
   mClause:                  STRING := '';
   mDetail1:                 STRING := '';
   mDetail2:                 STRING := '';
   Find:                     BOOLEAN := TRUE;
END_LOCAL;
   mRelProperties := QUERY (RelP <* Door\IfcObject.IsDefinedBy |
                             (RelP.RelatingPropertyDefinition.Name = 'Pset_DoorCommon'));
   IF (SIZEOF (mRelProperties) = 1)
   THEN
      pw1 := SIZEOF (mRelProperties[1].RelatingPropertyDefinition.HasProperties);
      REPEAT pw2 := 1 To pw1;
        IF ((mRelProperties[1].RelatingPropertyDefinition.HasProperties[pw2] IS IfcSimpleProperty)
        AND (mRelProperties[1].RelatingPropertyDefinition.HasProperties[pw2].Name = 'NominalWidth'))
        THEN
           mDoorWidth := mRelProperties[1].RelatingPropertyDefinition.HasProperties[pw2].ValueComponent;
           ESCAPE;
        END_IF;
      END_REPEAT;
   END_IF;
   IF (mDoorWidth < 800.0)
   THEN
      Find := FALSE;
      mRelContains := QUERY (RelC <* Door\IfcObject.IsContainedBy |
              (RelC.RelationshipType = SpaceContainer));
      IF (SIZEOF (mRelContains) >= 1)
      THEN
         mSpace := mRelContains[1].RelatingObject;
         mSpaceName := Get_Space_Name(mSpace);
         mSpaceRef :=  Get_Space_Ref(mSpace);
      END_IF;
      mDoorLabel :=  Get_Object_Label(Door);
```

Figure 4. An illustration of the process of encoding a clause of building codes to an object-based interpretation and then to an EDM rule schema.

CSM is a Java-implemented schema that defines a rule base displayed to the user as constraints. There are over 150 different constraints defined within the Constraint Libraries and Constraint Sets of CSM 1.0/1.1. Constraints can be divided into ten main groups containing unique constraints for constructing specialised rules, including: Construction Constraints, Legacy Constraints, A Precheck of the Model (CAD components), Checking for Quantity Take Off Purposes, Escape Routes, Interference Checking, Construction Type, Space Checking, Typical Modelling Errors and Visualisation Constraints, Figure 5.



Constraint Set Manager

File Tools Help

- Constraint Libraries
  - solibriConstructionConstraints 1
  - solibriLegacyConstraints 2
- Constraint Sets
  - A Precheck of the Model 3
  - Checking for Quantity Take-Off Purposes 4
  - Component Checking
  - Corporate Specific Constraints and Parameters
  - Escape Routes 5
  - Getting Started
  - Interference Checking 6
  - Interferences - Objects
  - ProIT Guidelines 7
  - Sample Check
  - Space Checking 8
  - Templates
  - Typical modeling errors 9
  - Visualization Constraints 10

Figure 5. An illustration of the constraint groups provided by CSM 1.1.

The capability of the CSM rule base to encode the performance requirements and domain knowledge from building codes was investigated. CSM allows structuring of new constraints to be composed from generic constraints available in libraries. Control of constraint behaviours is possible by setting parameter values for specific performance requirements from building codes. Figure 6 illustrates how SMC is employed for encoding building codes and operated in conjunction with SMC.

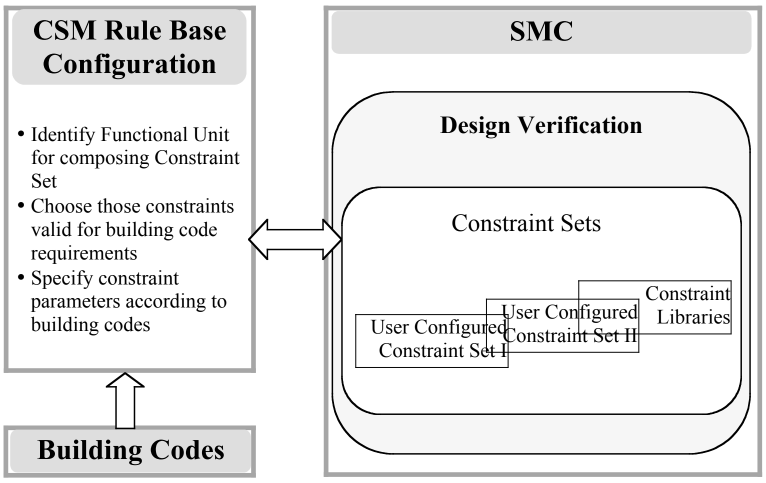

CSM Rule Base Configuration
• Identify Functional Unit for composing Constraint Set
• Choose those constraints valid for building code requirements
• Specify constraint parameters according to building codes
Building Codes
SMC
Design Verification
Constraint Sets
User Configured Constraint Set I
User Configured Constraint Set II
Constraint Libraries

Figure 6. An illustration of developing CSM rule base for code checking.

SMC defines highly focused constraints with the CSM for encoding design requirements and as a result, descriptions are inflexible. Some flexibility is provided by the ability to set parameter values, however, the parameters are narrow and not available for all components and across all types of constraints. When design verification requires an interpretation of the performance requirements, e.g. an interpretation of a complex building code clause where domain-specific knowledge is demanded, SMC lacks a mechanism for encoding such interpretations.

4. EDM PROTOTYPE SYSTEM

This project implemented an EDM prototype system based on a proposed framework illustrated in Figure 7. We used ArchiCAD as an example object-based CAD system and chose a small 3 storeys building from SMC for testing.

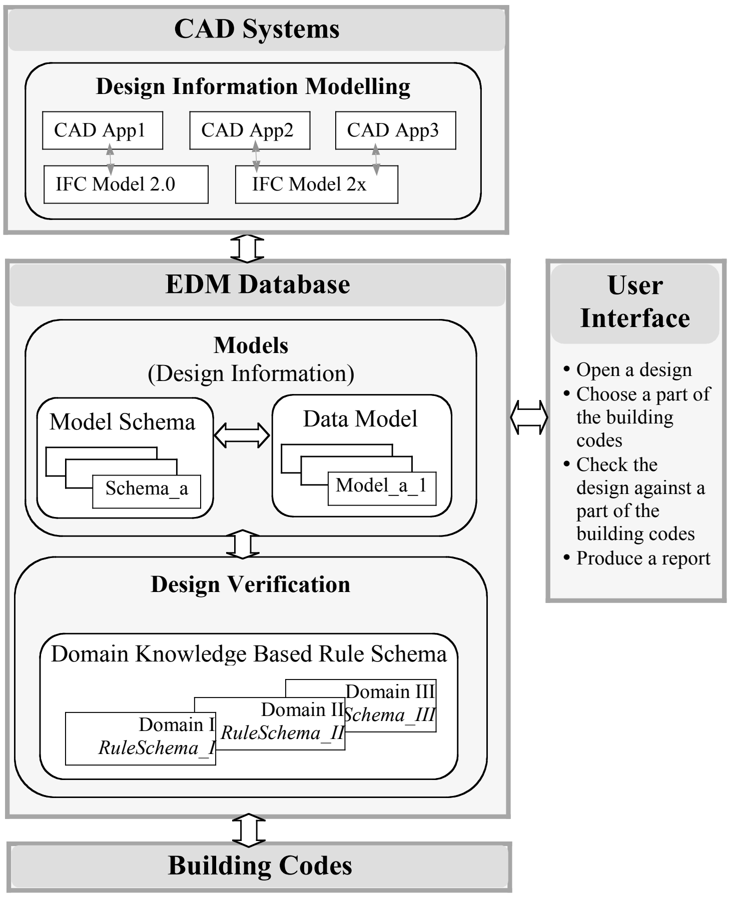


Figure 7. An illustration of a framework for the EDM prototype system.

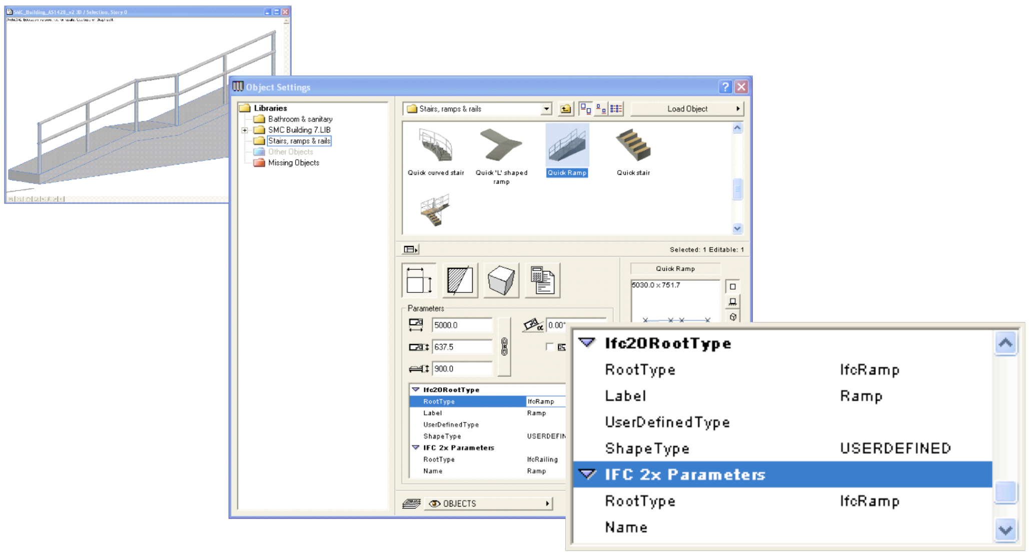
4.1 MODELLING DESIGN INFORMATION

The process of automating code checking requires that one begins with an adequate building model. ArchiCAD supports object-based information modelling while ArchiCAD IFC add-ons provide a way to define and export extensible and interoperable model data. This project uses the IFC conversion and property extension mechanisms to achieve extended design information for ArchiCAD objects. An example of converting an ArchiCAD object into an extensive IFC object is presented in Figure 8, where a Ramp object was converted into IfcRamp with extended information specified as properties.

Mappings between CAD, IFC and building code models require uniform object identification and description. The information requirements for a complete mapping schema between models are fundamental to the task of automating code checking. The following general requirements have been considered for such mappings:

- Defining uniform building objects ID descriptions;
- Defining semantics of object properties;

- Structuring extended objects and properties;
- Instantiating definitions and parameters.



```
Ifc20RootType
  RootType          IfcRamp
  Label             Ramp
  UserDefinedType
  ShapeType         USERDEFINED
IFC 2x Parameters
  RootType          IfcRamp
  Name
```

Figure 8. An example of converting an ArchiCAD object into an IFC object.

Table 1 outlines examples for object mappings between CAD, IFC and building codes models.

AS 1428.1 Building Object	ArchiCAD Element/ Object	Mapping to IFC Object
Walkway	Create Zone	IFC_SPACE
Ramp	Library Object = Ramp	IFC_RAMP
Kerb Ramp or Step Ramp	Create Object	IFC_RAMP
Landing (Stair or Ramp)	Create Object	IFC_STAIRFLIGHT or IFC_RAMPFLIGHT
Circulation	Create Zone	IFC_SPACE
Path of Travel	Create Zone	IFC_SPACE, IFC_DOOR
Door / Swing Door	Element = Door	IFC_DOOR_DBLSWING IFC_DOOR_SGLSWING
Sliding Door / Surface Mounted Door	Element = Door	IFC_DOOR_SLIDING
Automatic Door	Element = Door	IFC_DOOR_REVOLVING
Glazed Door	Element = Door	IFC_DOOR_DBLSWING IFC_DOOR_SGLSWING
Toilet Door	Element = Door	IFC_DOOR_SGLSWING
Doorway	Element = Door	IFC_DOOR, IFC_SPACE
Gate	Create Object	IFC_PROXY
Opening	Element = Opening	IFC_OPENINGELEMENT
Entrance Door	Element = Door	IFC_DOOR_EXTERIOR
Handrail	Library Object = Handrail	IFC_HANDRAIL

Table 1. Examples of object mappings between CAD, IFC and building codes models.

In order to provide a building model that tests AS 1428 Part 1 clauses, building elements are selected and re-modelled in ArchiCAD so as particular attributes and properties do not comply. Figure 9 presents the building model that we used for testing, where the illegal building elements are circled in red. For example, in the Lobby space in Figure 9, it shows that a revolving door is installed but there is no hinged or sliding door installed. This is not compliant with AS 1428 Part 1 Clause 7.1 (a).

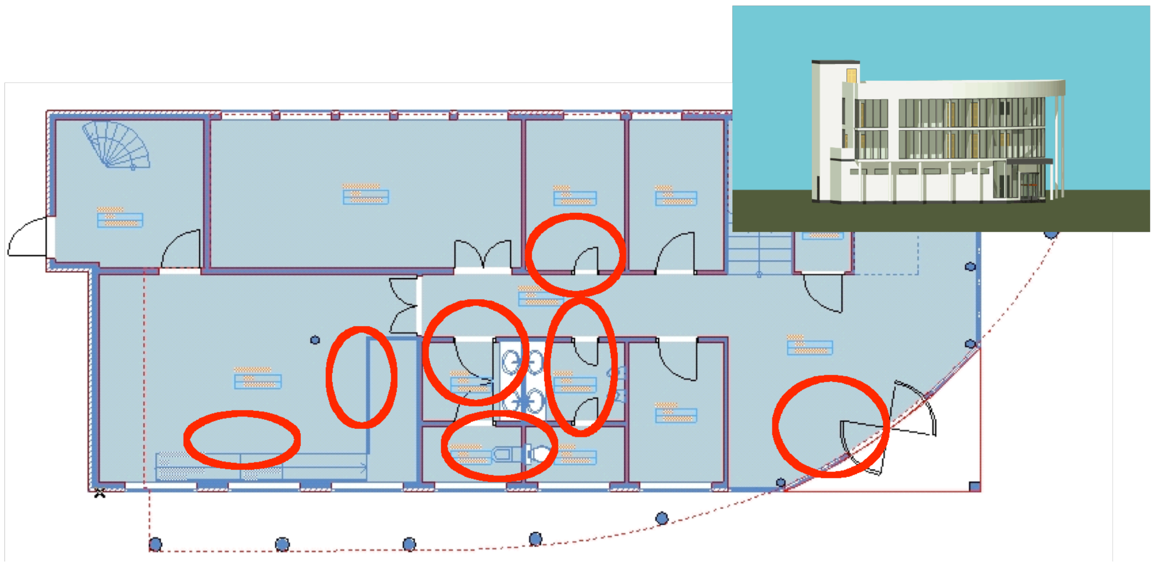


Figure 9. An example building model with Illegal building elements circled in red.

4.2 IMPLEMENTATION OF EDM DATABASE AND RULE BASE

An EDM database was created /opened to store the IFC-based building models from ArchiCAD. The building models consisting of object, properties and relationships were translated into the EDM database files.

There are seventeen clauses in AS 1428 Part 1. This project encoded the clauses where the objects and properties are supported by the current object-based CAD systems and IFC models. Table 2 presents an example of encoding a part of Clause 5 Walkways, Ramps and Landings.

Performance requirements in different areas are able to be encoded with a number of rule schemas that are supported by domain knowledge. Figure 10 illustrates a set of EDM schemas for encoding building codes in different areas such as fire safety, Roof and Wall Cladding, etc.

```
ENTITY IfcRamp;
WHERE
   Handrail:  Handrails_BothSides(self);
   Landing:   Landing_Length(self);
   Ramp:       Ramp_Width(self);
END_ENTITY;

FUNCTION Handrails_BothSides (Rmp: IfcRamp): LOGICAL;
LOCAL
   Find:                         BOOLEAN := FALSE;
   RampRelationProperties:       SET OF IfcRelAssignsProperties := [];
   cont1,cont2:            INTEGER;
   mObjValue:                    IfcMeasureValue;
   mObjName:                     STRING := ' ';
   mObjLabel:                    STRING := ' ';
   mObj Ref:                      STRING := 'N/A';
   mClause:                      STRING := ' ';
END_LOCAL;
   Find := True;
   RampRelProperties := QUERY (Ramp <* Rmp   \IfcObject.IsDefinedBy |
                             (Ramp.RelatingPropertyDefinition IS
                              IfcExtensionPropertySet));
   IF (SIZEOF (RampRelProperties) = 1)
   THEN
      cont1 := SIZEOF(RampRelProperties[1].
                      RelatingPropertyDefinition.HasProperties);
       REPEAT count2 := 1 To cont1;
          IF ((RampRelPropertie  s[1].
               RelatingPropertyDefinition.HasProperties[c1] IS
               IfcPropertyList)
               AND (RampRelProperties[1].
                   RelatingPropertyDefinition.HasProperties[c1].Name
                   = 'LIBPARAM'))
               THEN
                  --- ---
```

Table 2. An example of encoding a part of Clause 5 Walkways, Ramps and Landings.

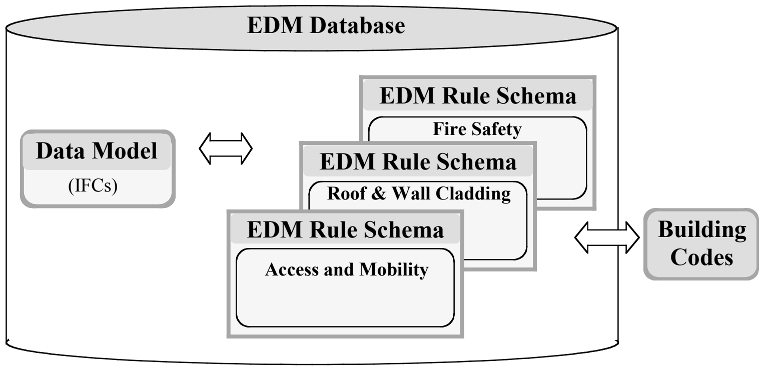


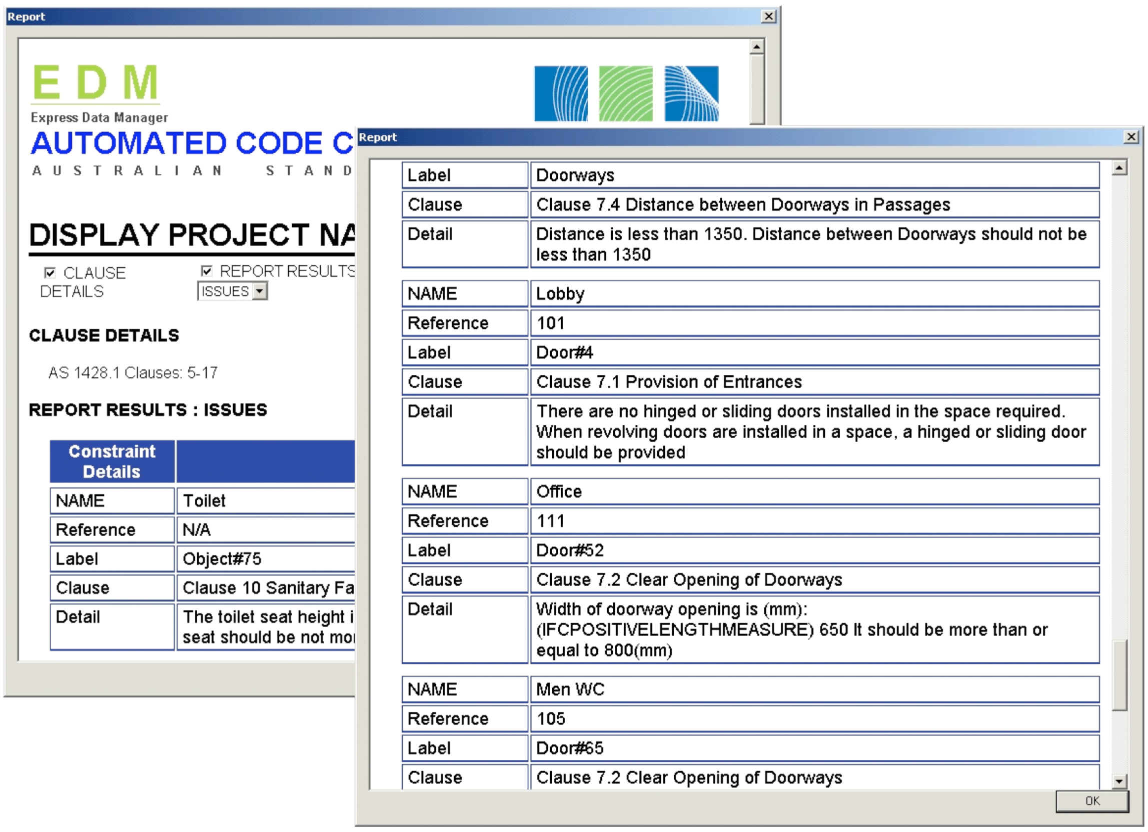
Figure 10. An example of encoding performance requirements in different areas with a set of EDM rule schemas.

4.3 EDM REPORTING SYSTEM

The EDM rule checking engine was tested for checking the building model against the new rule schema encoding clauses from AS 1428 Part 1. The system was designed to automatically generate a report listing the objects that were not compliant. The original report was in a text format without detailed features for viewing and analysing.

In this project, we developed a user friendly reporting system that allows detailed information and issues to be analysed. A new reporting schema was implemented with XML and HTML and interfaced to the EDM rule checking engine. Detailed issues involve: object identification, clause description and failed object features.

Figure 11 illustrates the issues from a typical run of checking the building design. We see the information for the Lobby object includes: a reference number of the Lobby object in ArchiCAD, identification for Door object contained in the Lobby that resulted in failure to comply, a description of Clause 7.4 and failed features.



DISPLAY PROJECT NA

☑ CLAUSE DETAILS ☑ REPORT RESULTS ISSUES

CLAUSE DETAILS

AS 1428.1 Clauses: 5-17

REPORT RESULTS : ISSUES

Constraint Details	
NAME	Toilet
Reference	N/A
Label	Object#75
Clause	Clause 10 Sanitary Fa
Detail	The toilet seat height i seat should be not mo

Label	Doorways
Clause	Clause 7.4 Distance between Doorways in Passages
Detail	Distance is less than 1350. Distance between Doorways should not be less than 1350
NAME	Lobby
Reference	101
Label	Door#4
Clause	Clause 7.1 Provision of Entrances
Detail	There are no hinged or sliding doors installed in the space required. When revolving doors are installed in a space, a hinged or sliding door should be provided
NAME	Office
Reference	111
Label	Door#52
Clause	Clause 7.2 Clear Opening of Doorways
Detail	Width of doorway opening is (mm): (IFCPOSITIVELENGTHMEASURE) 650 It should be more than or equal to 800(mm)
NAME	Men WC
Reference	105
Label	Door#65
Clause	Clause 7.2 Clear Opening of Doorways

Figure 11. Some issues from a typical run of checking designs.

5. SMC PROTOTYPE

A framework for implementing the SMC prototype is illustrated in Figure 12. Different from EDM, SMC provides ArchiCAD SMC add-ons so the building models in ArchiCAD can be converted into IFC models or directly exported into SMC models for design checking.

SMC provides a direct interface to ArchiCAD that will allow designers to modify designs in conjunction with compliance with building codes at real time.

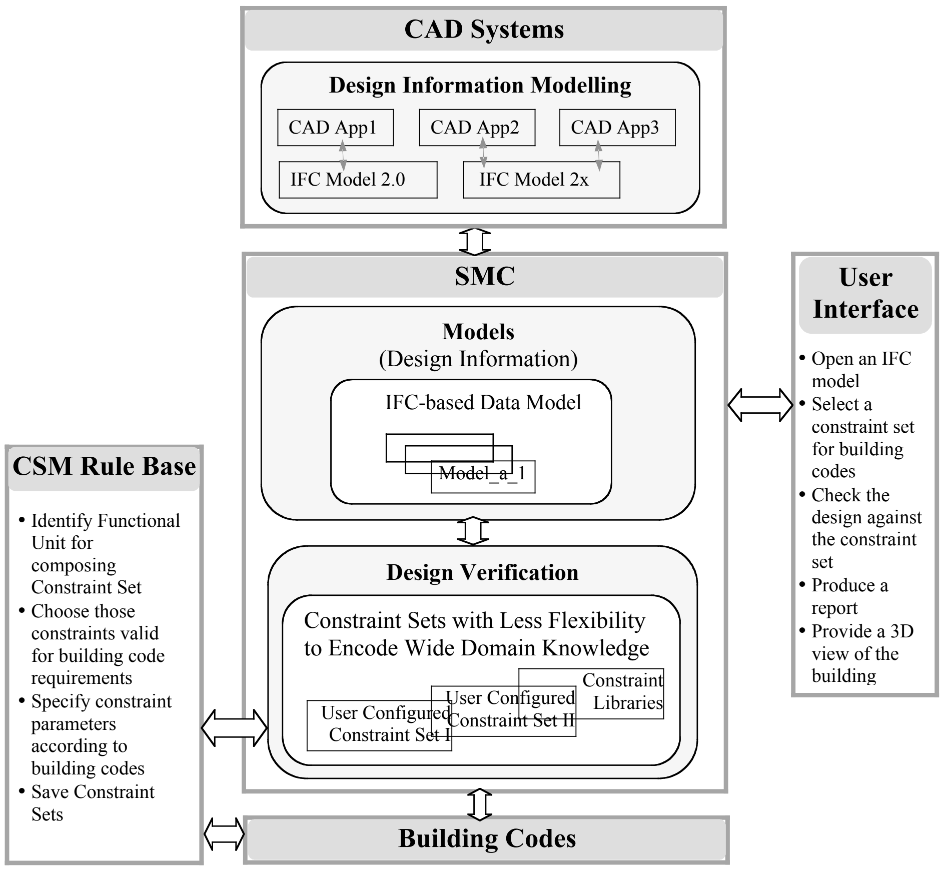

CAD Systems
Design Information Modelling
CAD App1
CAD App2
CAD App3
IFC Model 2.0
IFC Model 2x
SMC
Models
(Design Information)
IFC-based Data Model
Model_a_1
Design Verification
Constraint Sets with Less Flexibility to Encode Wide Domain Knowledge
User Configured Constraint Set I
User Configured Constraint Set II
Constraint Libraries
Building Codes
CSM Rule Base
• Identify Functional Unit for composing Constraint Set
• Choose those constraints valid for building code requirements
• Specify constraint parameters according to building codes
• Save Constraint Sets
User Interface
• Open an IFC model
• Select a constraint set for building codes
• Check the design against the constraint set
• Produce a report
• Provide a 3D view of the building

Figure 12. An illustration of a framework for the SMC prototype system.

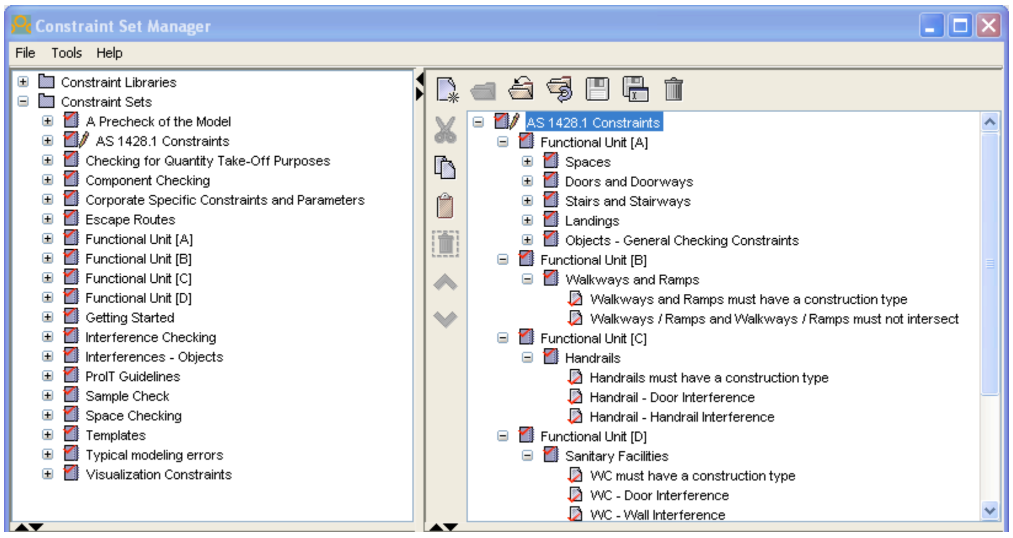
The SMC rule base provided by CSM is adequate for the purpose of checking how proficiently a building design is modelled in CAD systems and the precision in which various CAD objects have been represented. However, it shows less flexibility when used for the purpose of code checking.

5.1 IMPLEMENTATION OF SMC RULE BASE

A set of constraints were chosen from the SMC rule base for encoding clauses of AS 1428 Part1 and the constraint parameter values were set mapping onto specific performance requirements from AS 1428 Part 1 clauses. The following six groups were selected for mapping on to AS 1428 Part 1 building requirements:

- Construction Constraints;
- A precheck of the Model (CAD components);
- Checking for Quantity Take Off Purposes;
- Interference Checking;
- Construction Type;
- Spacing Checking.

These constraints were composed into Functional Units to check individual AS 1428 Part 1 building objects. Where CSM constraints map onto AS 1428 Part 1 clauses, the parameter values are adjusted to match the attributes of building objects. A detailed explanation of the constraint description and parameter specification is presented in Figure 13 (a) and (b).



- Constraint Libraries
- Constraint Sets
  - A Precheck of the Model
  - AS 1428.1 Constraints
  - Checking for Quantity Take-Off Purposes
  - Component Checking
  - Corporate Specific Constraints and Parameters
  - Escape Routes
  - Functional Unit [A]
  - Functional Unit [B]
  - Functional Unit [C]
  - Functional Unit [D]
  - Getting Started
  - Interference Checking
  - Interferences - Objects
  - ProIT Guidelines
  - Sample Check
  - Space Checking
  - Templates
  - Typical modeling errors
  - Visualization Constraints

- AS 1428.1 Constraints
  - Functional Unit [A]
    - Spaces
    - Doors and Doorways
    - Stairs and Stairways
    - Landings
    - Objects - General Checking Constraints
  - Functional Unit [B]
    - Walkways and Ramps
      - Walkways and Ramps must have a construction type
      - Walkways / Ramps and Walkways / Ramps must not intersect
  - Functional Unit [C]
    - Handrails
      - Handrails must have a construction type
      - Handrail - Door Interference
      - Handrail - Handrail Interference
  - Functional Unit [D]
    - Sanitary Facilities
      - WC must have a construction type
      - WC - Door Interference
      - WC - Wall Interference

(a)

4.3.1.1Space [A]

Spaces must have a name

✍	DESCRIPTION :	*This constraint checks that the name is set for all spaces in the model.*
✍	PARAMETERS:	*Space Property* - **Name** .

Space numbers must be unique

✍	DESCRIPTION:	*This constraint checks that there are no duplicate space numbers in the model.*
✍	PARAMETERS:	*Space Property* – **Space Number** .

Spaces must have an access

✍	DESCRIPTION :	*This constraint checks that each space can be accessed through a door.*
✍	PARAMETERS:	–

Spaces must not intersect other spaces

✍	DESCRIPTION:	*This constraint checks intersections of spaces.*
✍	PARAMETERS:	*Component types to be checked* - **Space** + **Space** .

Interference Type: ✍ *Duplicate*
✍ *Inside*
✍ *Overlappi ng*
Horizontal Interference : ✍ *Smaller Dimension* = **0.01***m*
Vertical Interfere nce: ✍ *Intrusion Height* = **3***m*

Airlock etc must not be too small

✍	DESCRIPTION:	*This constraint checks too small spaces.*
✍	PARAMETERS:	*Component Type* – **Space** .

Property Value Constraints: *Property* = **Width**

(b)

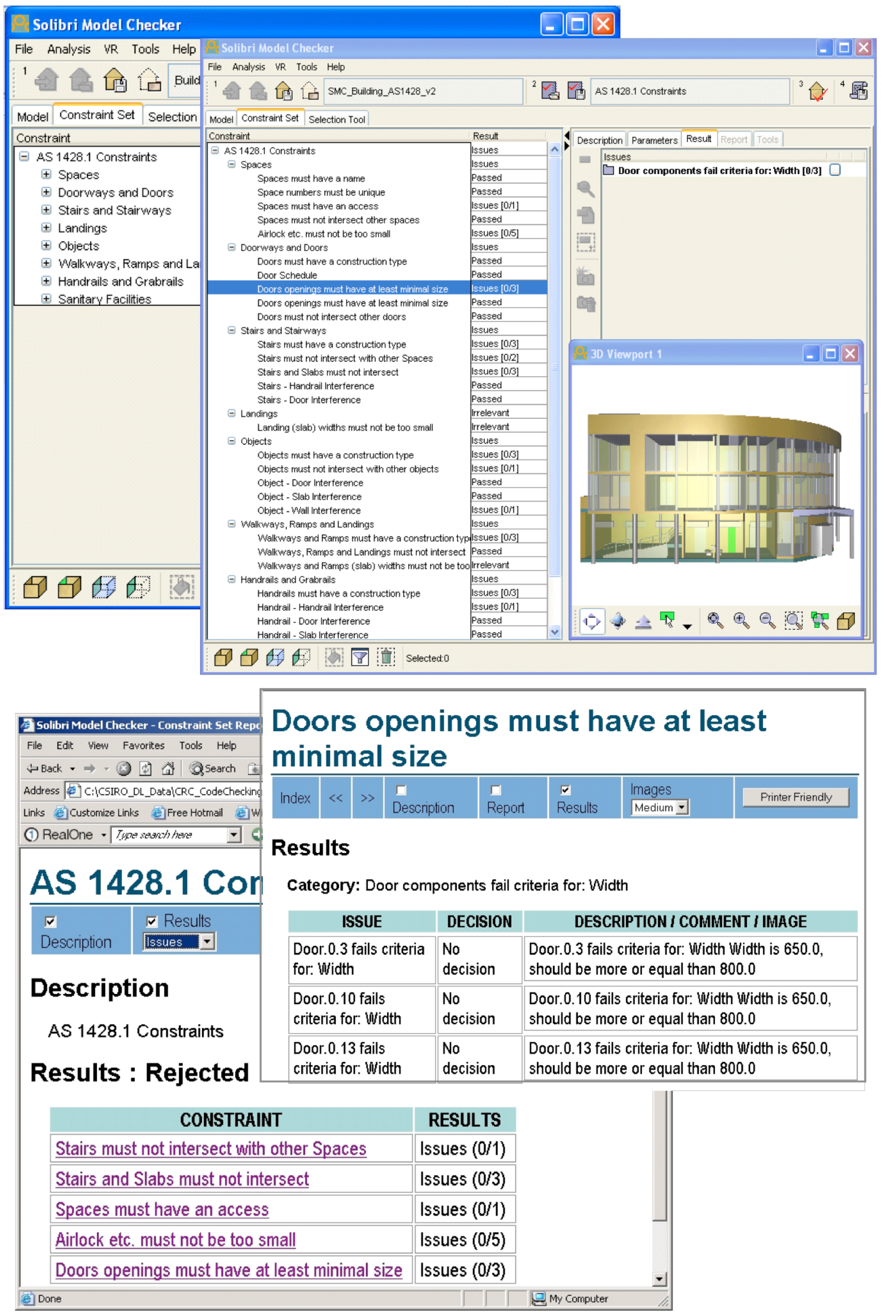
Figure 13. (a) An illustration of constraints composed into Functional Units to check individual AS 1428 Part 1 building objects and (b) A detailed explanation of the constraint description and parameter specification.

The SMC rule base does not contain constraints that check relations between building elements other than constraints for Interference Checking. Nor does it allow encoding of interpretations for performance requirements. The rule base can only be controlled by choosing among a number of building objects, library constraints and constraint parameter values. As a consequence, it has less flexibility to encode domain-specific knowledge.

5.2 SMC REPORTING SYSTEM

SMC provides a well developed reporting system with 3D visualization. The reporting interface allows designers to view the constraints being checked, update constraints as well as toggle between the reporting results and constraint specifications. Checking results can be grouped into four categories: All, Passed, Irrelevant and Issues.

For analysis purposes, the SMC reporting system allows: navigation of the building model from a 3D view port, identification and highlighting of the building elements that are not compliant as well as active links among reported issues, constraints, building components and their properties, Figure 14.


Solibri Model Checker
File Analysis VR Tools Help
SMC_Building_AS1428_v2
AS 1428.1 Constraints
Model Constraint Set Selection Tool
Description Parameters Result Report Tools
3D Viewport 1
Selected:0
Solibri Model Checker - Constraint Set Report
File Edit View Favorites Tools Help
Back Search
Address C:\CSIRO_DL_Data\CRC_CodeChecking
Links Customize Links Free Hotmail
RealOne Type search here
Index << >> Description Report Results Images Medium Printer Friendly
Done My Computer

Constraint
AS 1428.1 Constraints
- Spaces
- Doorways and Doors
- Stairs and Stairways
- Landings
- Objects
- Walkways, Ramps and La
- Handrails and Grabrails
- Sanitary Facilities

Constraint	Result
AS 1428.1 Constraints	Issues
Spaces	Issues
Spaces must have a name	Passed
Space numbers must be unique	Passed
Spaces must have an access	Issues [0/1]
Spaces must not intersect other spaces	Passed
Airlock etc. must not be too small	Issues [0/5]
Doorways and Doors	Issues
Doors must have a construction type	Passed
Door Schedule	Passed
Doors openings must have at least minimal size	Issues [0/3]
Doors openings must have at least minimal size	Passed
Doors must not intersect other doors	Passed
Stairs and Stairways	Issues
Stairs must have a construction type	Issues [0/3]
Stairs must not intersect with other Spaces	Issues [0/2]
Stairs and Slabs must not intersect	Issues [0/3]
Stairs - Handrail Interference	Passed
Stairs - Door Interference	Passed
Landings	Irrelevant
Landing (slab) widths must not be too small	Irrelevant
Objects	Issues
Objects must have a construction type	Issues [0/3]
Objects must not intersect with other objects	Issues [0/1]
Object - Door Interference	Passed
Object - Slab Interference	Passed
Object - Wall Interference	Issues [0/1]
Walkways, Ramps and Landings	Issues
Walkways and Ramps must have a construction typ	Issues [0/3]
Walkways, Ramps and Landings must not intersect	Passed
Walkways and Ramps (slab) widths must not be too	Irrelevant
Handrails and Grabrails	Issues
Handrails must have a construction type	Issues [0/3]
Handrail - Handrail Interference	Issues [0/1]
Handrail - Door Interference	Passed
Handrail - Slab Interference	Passed

Issues
Door components fail criteria for: Width [0/3]

Doors openings must have at least minimal size

Results

Category: Door components fail criteria for: Width

ISSUE	DECISION	DESCRIPTION / COMMENT / IMAGE
Door.0.3 fails criteria for: Width	No decision	Door.0.3 fails criteria for: Width Width is 650.0, should be more or equal than 800.0
Door.0.10 fails criteria for: Width	No decision	Door.0.10 fails criteria for: Width Width is 650.0, should be more or equal than 800.0
Door.0.13 fails criteria for: Width	No decision	Door.0.13 fails criteria for: Width Width is 650.0, should be more or equal than 800.0

AS 1428.1 Con

Description Results Issues

Description

AS 1428.1 Constraints

Results : Rejected

CONSTRAINT	RESULTS
Stairs must not intersect with other Spaces	Issues (0/1)
Stairs and Slabs must not intersect	Issues (0/3)
Spaces must have an access	Issues (0/1)
Airlock etc. must not be too small	Issues (0/5)
Doors openings must have at least minimal size	Issues (0/3)

Figure 14. An illustration of the issues from the SMC reporting system.

6. FUTURE DEVELOPMENT

6.1 A SUMMARY OF EDM AND SMC PROTOYPES

This section summarises the EDM prototype and SMC prototype through a comparative analysis.

The EDM prototype shows flexibility in modelling extended design information and encoding of wider domain-specific knowledge. The EDM database allows definitions of own model schema for building models that contain comprehensive design information as well as identical descriptions that map more directly onto building codes.

Compared with SMC, EDM doesn't provide a direct interface to CAD systems. However, it can be developed through the Application Program Interface of CAD systems when necessary.

The EDM prototype demonstrates an automated checking process comprising: importing building models into the EDM database, checking building models against defined EDM rule schema and reporting a list of the objects that failed to compliance. However, there is no user friendly reporting system for analysing issues. The system also lacks a user friendly interface for designers to monitor the information flows.

In testing the SMC prototype, the SMC rule base proved inadequate for checking building regulations and specific building codes since constraints were not able to encode design requirements at the required level.

SMC 2.0 has increased its component capability by including Stair object and general Objects for Interference and Construction Type checking. However, the parameters for Stair object and the general Objects do not support the range of checking procedures defined for existing components. In general, the SMC building model shows less flexibility in extending objects properties as well as mapping descriptions onto building codes.

The SMC prototype also demonstrates an automated checking process. It has shown its advantage of directly interfacing to CAD systems and excellent performance from its reporting system.

From the point view of application development, the EDM prototype provides a central object-based database containing a common model, which facilitates data sharing and communication with other applications. The SMC prototype provides building models based on a specific version of IFC. When the IFC standard is updated, the SMC models must also be upgraded.

6.2 AN APPROACH USING A SHARED OBJECT-BASED DATABASE

Since the EDM prototype provides more flexibility and capability for automating code checking, we consider this system as a focus for future development.

The EDM approach is able to provide a shared object-based database to contain design information and domain-specific knowledge. The EDM database has the Standard Data Interfaces, early /late bindings in Java, C++, XML, etc. to support data exchange and information communication with various applications including CAD applications.

The future development of the EDM approach lies in extending its ability toward defining extensive design information into the EDM Data Model as well as interpretations of performance requirements and domain knowledge into the EDM General Rule Base, Figure 15. This will fundamentally support the design verification required by automated code checking. The EDM Data Model will be defined to contain adequate design information mapping onto building codes. The EDM General Rule Base will be developed to encode domain- knowledge for general interpretations of design performances. Once specific criterion from building codes are updated, it is not therefore necessary to modify the whole EDM rule base.

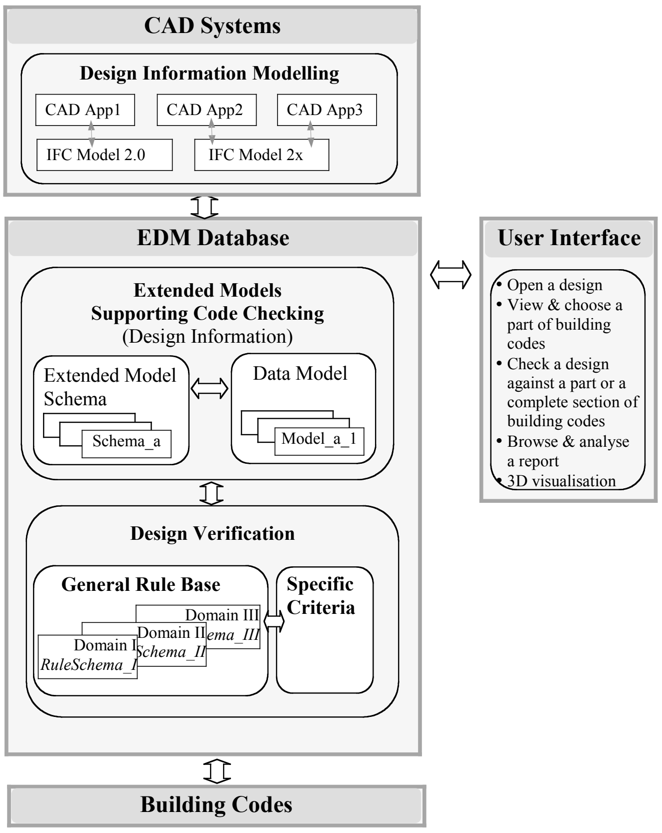


Figure 15. An illustration of the EDM approach for future development.

EDM's approach to the checking process will be improved through considering different levels of design checking in conjunction with different stages of design and different focuses on building objects or clauses of building codes.

6.3 POTENTIAL APPLICATIONS

This project can be applied in other potential areas in the construction industries. For example, when domain-specific knowledge in relation to project management or fire safety assessment is encoded to the EDM rule base, it can then be applied to check a building model against project management or fire risk analysis.

From this project, a prototype research for automated code checking has been completed. Future works will be towards improving the prototype for applications in AEC in Australia.

REFERENCES

Balachandran M., Rosenman M. A. and Gero J. S. (1991) A Knowledge-based Approach to the Automatic Verification of Designs from CAD Databases, in Gero J. S. (ed.), Artificial Intelligence in Design '91, Butterworth-Heinemann, Oxford, pp. 757-781.

Eastman C. M. (1999) Building Product Models: Computer Environments Supporting Design and Construction, CRC Press.

EPM Technology: (2002) EDMAssist4.5, vols 1-5, Norway: EPM Technology.

Gero J. S. (1982) A self-checking database for the Australian Building Code, CAD82, Butterworths, Guildford, pp. 119-125.

Graphisoft. ArchiCAD7.0 Graphisoft.

Graphisoft (2001) ArchiCAD IFC Reference Guide, Version 1.0, Graphisoft.

IAI (2000) IFC Specifications, www.iai-international.org.

IAI (2001) IFC 2x Model Implementation Guide, IAI.

Maissa S., Frachet J. P., Lombardo J. C., Bourdeau M. and Soubra S. (2002) Regulation Checking in a Virtual Building, CIB w78 conference 2002.

Rosenman M. A., Gero J. S. and Oxman R. (1986) An expert system for design codes and design rules, in Sriram D. and Adey R. (eds), Applications of Artificial Intelligence to Engineering Problems, Springer-Verlag, Berlin, pp. 745-758.

Stouffs R. and Krishnamurti R. (2001) On the Road to Standardization, in Proceedings of CAAD Futures 2001, Netherlands, 75-88.

Woodbury R., Burrow A., Drogemuller R. and Datta S. (2000) Code Checking by Representation Comparison, CAADRIA: Proceedings of the 5th Conference on Computer Aided Architectural Design Research in Asia, pp. 235-244, CASA, Singapore.

Yang Q. and Cui L. (2003) Interoperable and Extensible Information Modelling, CAADFutures 2003, Taiwan.