The Use of Smart Graphics for the Redevelopment of Historical Structures

by

SHAUNA M. HILL

A thesis submitted to the Department of Architecture and Graduate Studies of the University of Manitoba in partial fulfillment of the requirements for the degree of

MASTER OF ARCHITECTURE

ISBN 0-315-99052-X

Name

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

ARCHITECTURE — SUBJECT TERM

0 7 2 9 — SUBJECT CODE

U·M·I

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS

Architecture	0729
Art History	0377
Cinema	0900
Dance	0378
Fine Arts	0357
Information Science	0723
Journalism	0391
Library Science	0399
Mass Communications	0708
Music	0413
Speech Communication	0459
Theater	0465

EDUCATION

General	0515
Administration	0514
Adult and Continuing	0516
Agricultural	0517
Art	0273
Bilingual and Multicultural	0282
Business	0688
Community College	0275
Curriculum and Instruction	0727
Early Childhood	0518
Elementary	0524
Finance	0277
Guidance and Counseling	0519
Health	0680
Higher	0745
History of	0520
Home Economics	0278
Industrial	0521
Language and Literature	0279
Mathematics	0280
Music	0522
Philosophy of	0998
Physical	0523
Psychology	0525
Reading	0535
Religious	0527
Sciences	0714
Secondary	0533
Social Sciences	0534
Sociology of	0340
Special	0529
Teacher Training	0530
Technology	0710
Tests and Measurements	0288
Vocational	0747

LANGUAGE, LITERATURE AND LINGUISTICS

Language	
General	0679
Ancient	0289
Linguistics	0290
Modern	0291
Literature	
General	0401
Classical	0294
Comparative	0295
Medieval	0297
Modern	0298
African	0316
American	0591
Asian	0305
Canadian (English)	0352
Canadian (French)	0355
English	0593
Germanic	0311
Latin American	0312
Middle Eastern	0315
Romance	0313
Slavic and East European	0314

PHILOSOPHY, RELIGION AND THEOLOGY

Philosophy	0422
Religion	
General	0318
Biblical Studies	0321
Clergy	0319
History of	0320
Philosophy of	0322
Theology	0469

SOCIAL SCIENCES

American Studies	0323
Anthropology	
Archaeology	0324
Cultural	0326
Physical	0327
Business Administration	
General	0310
Accounting	0272
Banking	0770
Management	0454
Marketing	0338
Canadian Studies	0385
Economics	
General	0501
Agricultural	0503
Commerce-Business	0505
Finance	0508
History	0509
Labor	0510
Theory	0511
Folklore	0358
Geography	0366
Gerontology	0351
History	
General	0578
Ancient	0579
Medieval	0581
Modern	0582
Black	0328
African	0331
Asia, Australia and Oceania	0332
Canadian	0334
European	0335
Latin American	0336
Middle Eastern	0333
United States	0337
History of Science	0585
Law	0398
Political Science	
General	0615
International Law and Relations	0616
Public Administration	0617
Recreation	0814
Social Work	0452
Sociology	
General	0626
Criminology and Penology	0627
Demography	0938
Ethnic and Racial Studies	0631
Individual and Family Studies	0628
Industrial and Labor Relations	0629
Public and Social Welfare	0630
Social Structure and Development	0700
Theory and Methods	0344
Transportation	0709
Urban and Regional Planning	0999
Women's Studies	0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES

Agriculture	
General	0473
Agronomy	0285
Animal Culture and Nutrition	0475
Animal Pathology	0476
Food Science and Technology	0359
Forestry and Wildlife	0478
Plant Culture	0479
Plant Pathology	0480
Plant Physiology	0817
Range Management	0777
Wood Technology	0746
Biology	
General	0306
Anatomy	0287
Biostatistics	0308
Botany	0309
Cell	0379
Ecology	0329
Entomology	0353
Genetics	0369
Limnology	0793
Microbiology	0410
Molecular	0307
Neuroscience	0317
Oceanography	0416
Physiology	0433
Radiation	0821
Veterinary Science	0778
Zoology	0472
Biophysics	
General	0786
Medical	0760

EARTH SCIENCES

Biogeochemistry	0425
Geochemistry	0996
Geodesy	0370
Geology	0372
Geophysics	0373
Hydrology	0388
Mineralogy	0411
Paleobotany	0345
Paleoecology	0426
Paleontology	0418
Paleozoology	0985
Palynology	0427
Physical Geography	0368
Physical Oceanography	0415

HEALTH AND ENVIRONMENTAL SCIENCES

Environmental Sciences	0768
Health Sciences	
General	0566
Audiology	0300
Chemotherapy	0992
Dentistry	0567
Education	0350
Hospital Management	0769
Human Development	0758
Immunology	0982
Medicine and Surgery	0564
Mental Health	0347
Nursing	0569
Nutrition	0570
Obstetrics and Gynecology	0380
Occupational Health and Therapy	0354
Ophthalmology	0381
Pathology	0571
Pharmacology	0419
Pharmacy	0572
Physical Therapy	0382
Public Health	0573
Radiology	0574
Recreation	0575
Speech Pathology	0460
Toxicology	0383
Home Economics	0386

PHYSICAL SCIENCES

Pure Sciences

Chemistry	
General	0485
Agricultural	0749
Analytical	0486
Biochemistry	0487
Inorganic	0488
Nuclear	0738
Organic	0490
Pharmaceutical	0491
Physical	0494
Polymer	0495
Radiation	0754
Mathematics	0405
Physics	
General	0605
Acoustics	0986
Astronomy and Astrophysics	0606
Atmospheric Science	0608
Atomic	0748
Electronics and Electricity	0607
Elementary Particles and High Energy	0798
Fluid and Plasma	0759
Molecular	0609
Nuclear	0610
Optics	0752
Radiation	0756
Solid State	0611
Statistics	0463

Applied Sciences

Applied Mechanics	0346
Computer Science	0984
Engineering	
General	0537
Aerospace	0538
Agricultural	0539
Automotive	0540
Biomedical	0541
Chemical	0542
Civil	0543
Electronics and Electrical	0544
Heat and Thermodynamics	0348
Hydraulic	0545
Industrial	0546
Marine	0547
Materials Science	0794
Mechanical	0548
Metallurgy	0743
Mining	0551
Nuclear	0552
Packaging	0549
Petroleum	0765
Sanitary and Municipal	0554
System Science	0790
Geotechnology	0428
Operations Research	0796
Plastics Technology	0795
Textile Technology	0994

PSYCHOLOGY

General	0621
Behavioral	0384
Clinical	0622
Developmental	0620
Experimental	0623
Industrial	0624
Personality	0625
Physiological	0989
Psychobiology	0349
Psychometrics	0632
Social	0451

Nom ____

Dissertation Abstracts International est organisé en catégories de sujets. Veuillez s.v.p. choisir le sujet qui décrit le mieux votre thèse et inscrivez le code numérique approprié dans l'espace réservé ci-dessous.

____ ☐☐☐☐ U·M·I

SUJET CODE DE SUJET

Catégories par sujets

HUMANITÉS ET SCIENCES SOCIALES

COMMUNICATIONS ET LES ARTS

Architecture	0729
Beaux-arts	0357
Bibliothéconomie	0399
Cinéma	0900
Communication verbale	0459
Communications	0708
Danse	0378
Histoire de l'art	0377
Journalisme	0391
Musique	0413
Sciences de l'information	0723
Théâtre	0465

ÉDUCATION

Généralités	515
Administration	0514
Art	0273
Collèges communautaires	0275
Commerce	0688
Économie domestique	0278
Éducation permanente	0516
Éducation préscolaire	0518
Éducation sanitaire	0680
Enseignement agricole	0517
Enseignement bilingue et multiculturel	0282
Enseignement industriel	0521
Enseignement primaire.	0524
Enseignement professionnel	0747
Enseignement religieux	0527
Enseignement secondaire	0533
Enseignement spécial	0529
Enseignement supérieur	0745
Évaluation	0288
Finances	0277
Formation des enseignants	0530
Histoire de l'éducation	0520
Langues et littérature	0279
Lecture	0535
Mathématiques	0280
Musique	0522
Orientation et consultation	0519
Philosophie de l'éducation	0998
Physique	0523
Programmes d'études et enseignement	0727
Psychologie	0525
Sciences	0714
Sciences sociales	0534
Sociologie de l'éducation	0340
Technologie	0710

LANGUE, LITTÉRATURE ET LINGUISTIQUE

Langues	
Généralités	0679
Anciennes	0289
Linguistique	0290
Modernes	0291
Littérature	
Généralités	0401
Anciennes	0294
Comparée	0295
Mediévale	0297
Moderne	0298
Africaine	0316
Américaine	0591
Anglaise	0593
Asiatique	0305
Canadienne (Anglaise)	0352
Canadienne (Française)	0355
Germanique	0311
Latino-américaine	0312
Moyen-orientale	0315
Romane	0313
Slave et est-européenne	0314

PHILOSOPHIE, RELIGION ET THÉOLOGIE

Philosophie	0422
Religion	
Généralités	0318
Clergé	0319
Études bibliques	0321
Histoire des religions	0320
Philosophie de la religion	0322
Théologie	0469

SCIENCES SOCIALES

Anthropologie	
Archéologie	0324
Culturelle	0326
Physique	0327
Droit	0398
Économie	
Généralités	0501
Commerce-Affaires	0505
Économie agricole	0503
Économie du travail	0510
Finances	0508
Histoire	0509
Théorie	0511
Études américaines	0323
Études canadiennes	0385
Études féministes	0453
Folklore	0358
Géographie	0366
Gérontologie	0351
Gestion des affaires	
Généralités	0310
Administration	0454
Banques	0770
Comptabilité	0272
Marketing	0338
Histoire	
Histoire générale	0578
Ancienne	0579
Médiévale	0581
Moderne	0582
Histoire des noirs	0328
Africaine	0331
Canadienne	0334
États-Unis	0337
Européenne	0335
Moyen-orientale	0333
Latino-américaine	0336
Asie, Australie et Océanie	0332
Histoire des sciences	0585
Loisirs	0814
Planification urbaine et régionale	0999
Science politique	
Généralités	0615
Administration publique	0617
Droit et relations internationales	0616
Sociologie	
Généralités	0626
Aide et bien-àtre social	0630
Criminologie et établissements pénitentiaires	0627
Démographie	0938
Études de l' individu et de la famille	0628
Études des relations interethniques et des relations raciales	0631
Structure et développement social	0700
Théorie et méthodes.	0344
Travail et relations industrielles	0629
Transports	0709
Travail social	0452

SCIENCES ET INGÉNIERIE

SCIENCES BIOLOGIQUES

Agriculture	
Généralités	0473
Agronomie.	0285
Alimentation et technologie alimentaire	0359
Culture	0479
Élevage et alimentation	0475
Exploitation des péturages	0777
Pathologie animale	0476
Pathologie végétale	0480
Physiologie végétale	0817
Sylviculture et faune	0478
Technologie du bois	0746
Biologie	
Généralités	0306
Anatomie	0287
Biologie (Statistiques)	0308
Biologie moléculaire	0307
Botanique	0309
Cellule	0379
Écologie	0329
Entomologie	0353
Génétique	0369
Limnologie	0793
Microbiologie	0410
Neurologie	0317
Océanographie	0416
Physiologie	0433
Radiation	0821
Science vétérinaire	0778
Zoologie	0472
Biophysique	
Généralités	0786
Medicale	0760

SCIENCES DE LA TERRE

Biogéochimie	0425
Géochimie	0996
Géodésie	0370
Géographie physique	0368
Géologie	0372
Géophysique	0373
Hydrologie	0388
Minéralogie	0411
Océanographie physique	0415
Paléobotanique	0345
Paléoécologie	0426
Paléontologie	0418
Paléozoologie	0985
Palynologie	0427

SCIENCES DE LA SANTÉ ET DE L'ENVIRONNEMENT

Économie domestique	0386
Sciences de l'environnement	0768
Sciences de la santé	
Généralités	0566
Administration des hipitaux	0769
Alimentation et nutrition	0570
Audiologie	0300
Chimiothérapie	0992
Dentisterie	0567
Développement humain	0758
Enseignement	0350
Immunologie	0982
Loisirs	0575
Médecine du travail et thérapie	0354
Médecine et chirurgie	0564
Obstétrique et gynécologie	0380
Ophtalmologie	0381
Orthophonie	0460
Pathologie	0571
Pharmacie	0572
Pharmacologie	0419
Physiothérapie	0382
Radiologie	0574
Santé mentale	0347
Santé publique	0573
Soins infirmiers	0569
Toxicologie	0383

SCIENCES PHYSIQUES

Sciences Pures

Chimie	
Genéralités	0485
Biochimie	487
Chimie agricole	0749
Chimie analytique	0486
Chimie minérale	0488
Chimie nucléaire	0738
Chimie organique	0490
Chimie pharmaceutique	0491
Physique	0494
PolymÇres	0495
Radiation	0754
Mathématiques	0405
Physique	
Genéralités	0605
Acoustique	0986
Astronomie et astrophysique	0606
Electronique et électricité	0607
Fluides et plasma	0759
Météorologie	0608
Optique	0752
Particules (Physique nucléaire)	0798
Physique atomique	0748
Physique de l'état solide	0611
Physique moléculaire	0609
Physique nucléoire	0610
Radiation	0756
Statistiques	0463

Sciences Appliqués Et Technologie

Informatique	0984
Ingénierie	
Généralités	0537
Agricole	0539
Automobile	0540
Biomédicale	0541
Chaleur et ther modynamique	0348
Conditionnement (Emballage)	0549
Génie aérospatial	0538
Génie chimique	0542
Génie civil	0543
Génie électronique et électrique	0544
Génie industriel	0546
Génie mécanique	0548
Génie nucléaire	0552
Ingénierie des systòmes	0790
Mécanique navale	0547
Métallurgie	0743
Science des matériaux	0794
Technique du pétrole	0765
Technique minière	0551
Techniques sanitaires et municipales	0554
Technologie hydraulique	0545
Mécanique appliquée	0346
Géotechnologie	0428
Matières plastiques (Technologie)	0795
Recherche opérationnelle	0796
Textiles et tissus (Technologie)	0794

PSYCHOLOGIE

Généralités	0621
Personnalité	0625
Psychobiologie	0349
Psychologie clinique	0622
Psychologie du comportement	0384
Psychologie du développement	0620
Psychologie expérimentale	0623
Psychologie industrielle	0624
Psychologie physiologique	0989
Psychologie sociale	0451
Psychométrie	0632

PERMISSION TO QUOTE / REPRODUCE COPYWRITED MATERIAL

I (We,) **M. De Waard, F. Tolman**, owner(s) of the copyright to the work known as:

"MODELLING OF BUILDING REGULATIONS"

(VTT Symposium, Espoo, Finland, 27-29 May 1991)

hereby authorize Shauna M. Hill to use the following material as part of her master's thesis to be submitted to the University of Manitoba.

Page	Inclusive line numbers	Beginning & ending words or other identification
198	Figure 4	"View of article 41 on the product..."

I (We) futher extend this authorization to the University Microfilms International, National Library of Canada, for the purposes of reproducing and distributing microformed copies of the thesis.

[illegible]
Signature

SENT BY:Building Performance : 9-26-94 : 2:17PM : FAX :# 1/

Post-It™ brand fax transmittal memo 7671 | # of pages ▸ 1

To	From
Co.	Co.
Dept.	Phone #
Fax #	Fax #

PERMISSION TO QUOTE / REPRODUCE COPYWRITED MATERIAL

I (We,) **S. Cornick, D. Leishman, J.R. Thomas**, owner(s) of the copyright to the work known as:

"INTEGRATING BUILDING CODES INTO DESIGN SYSTEMS"

(VTT Symposium, Espoo, Finland, 27-29 May 1991)

hereby authorize Shauna M. Hill to use the following material as part of her master's thesis to be submitted to the University of Manitoba.

Page	Inclusive line numbers	Beginning & ending words or other identification
225	Figure 6a	"A simplified directed graph..."

I (We) futher extend this authorization to the University Microfilms International, National Library of Canada, for the purposes of reproducing and distributing microformed copies of the thesis.

Signature

THE USE OF SMART GRAPHICS FOR THE REDEVELOPMENT OF HISTORICAL STRUCTURES

BY

SHAUNA M. HILL

A Thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements for the degree of

MASTER OF ARCHITECTURE

This work is dedicated to my parents,
Norman and Elizabeth Hill,
who make all things possible
and to my *friends of many years*
for their tremendous support and encouragement.

ACKNOWLEDGMENTS

The author wishes to express her sincere appreciation to a number of persons who have contributed their valuable time, expertise, and support to her research. Firstly she would like to acknowledge her thesis committee: Professors Thompson and Perron, Mr. Tom Dixon, and Mr. Norman Blackie, for their assistance and advice throughout the course of this research and in the preparation of this manuscript. Robin Letellier, Chief of the Heritage Recording and Technical Data Services with Government Services Canada (Ottawa) is recognized for providing information about digital heritage building records along with Guy Masson, a conservation architect with Parks Canada (Prairie and Northern Region), for his help in identifying issues and sources of information on the topic of redevelopment of heritage buildings. John Frye, Head of the Plan Examination Branch with the City of Winnipeg, kindly provided the expert knowledge encoded the prototype knowledge base systems. Sincere and special thanks to Dr. Mark Evans and Linda Strachan from the Department of Computer Science for providing technical and programming support in the creation of the prototype computer systems.

The author would also like to recognize the financial contribution towards the presentation of this work at several national and international conferences from the President's Office of the University of Manitoba, Faculty of Architecture Endowment Fund, Dean's Office of the Faculty of Architecture, Graduate Studies, and the Department of Computer Science.

ABSTRACT OF THESIS

THE USE OF SMART GRAPHICS
FOR THE REDEVELOPMENT OF HISTORICAL STRUCTURES

by Shauna M. Hill

The redevelopment of historical structures is particularly difficult because it involves capturing and managing large amounts of existing building data and analysis of that data using specialized expertise from a variety of disciplines. With the advent of affordable and powerful computer-based information technologies researchers are becoming interested in developing design decision-making support systems capable of providing designers with access to both digital building data and special expertise. The Smart Graphic Environment developed and evaluated in this thesis is a type of Intelligent Computer Aided Design [ICAD] or Knowledge Assisted Design [KAD] system that integrates a commercially available CAD system with expertise encoded in a Knowledge Base (expert) System. The purpose of this environment is to provide designers working with heritage structures access to specialized expertise from within the CAD environment for improved design decision-making.

This thesis research includes an investigation of existing research in this area and the development of a conceptual framework for a Smart Graphic Environment. Object-based techniques for design data and knowledge acquisition and representation on the computer are explained. The conceptual framework was tested and evaluated through the implementation of a working prototype system This system analyses CAD files for building code violations.

The findings of this research identify the potential of such environments for information management, increasing productivity, and to improving the quality of design decision-making. One of the observations of this study is that because of current limitations of CAD models and knowledge representation the principle role of such computer systems is to augment rather than replace the skill and expertise of a human designer.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure		Page No.

Figure Page No.

LIST OF ABBREVIATIONS

Abbreviation	Long Form
ASCII	American Standard Code for Information Interchange
AAC	Architectural Attribute Coding
AI	Artificial Intelligence
AIA	American Institute of Architects
BOMA	Building Owners and Managers Association
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
CSA	Canadian Standards Association
DBMS	Database Management System
DIPAD	Digital Phototgrammetry and Architectural Design system
DPI	Dots Per Inch
DXF	Drawing Exchange Format
GIS	Geographic Information System
HABS	Historic Architectural Building Survey
HAER	Historic Engineering Record
HRMS	Historic Recording Management System
ICADs	Intelligent Computer Aided Design System
IGDB	Integrated Graphic Database
KAD	Knowledge Assisted Design
KB	Knowledge Base
KBS	Knowledge Base System
LISP	List Processing
NBCC	National Building Code of Canada
NIAM	Nijessen Information Analysis Method
OCR	Optical Character Reader
SGI	Scaled Graphic Information

Abbreviation	Long Form
SGE	Smart Graphic Environment
TTI	Text and Tabular Information
UGI	Unscaled Graphic Information
WES	Window Expert System

CHAPTER I

INTRODUCTION

I.A. Introduction

In 1624 Sir Henry Wotton wrote, "Well building hath three Conditions: Commoditie, Firmenes, and Delight" (1). Although these conditions are as desirable today as they were in 1624, one wonders if Sir Wotton could ever have imagined a time when architects would be working to achieve them in the *Information Age*. The availability of inexpensive, faster and ever more powerful computers has resulted in a social and technological revolution of global proportions. The impact on design professionals is that now they must learn how to deal with problems of creating, storing, retrieving, and managing a virtual tidal wave of digital information while at the same time be able to predict and maintain building performance criteria. Within today's context an increasing amount of special expertise is required to achieve the cost-effective, efficient, and code compliant buildings that modern clients demand.

In the 1990's Architects are more likely to be familiar with the catch phrase "reduce, reuse, and recycle." One of the fastest growing sectors of the construction industry is rehabilitation of existing structures. Today it makes up nearly half of the construction dollars spent. By 1998 the rehabilitation industry in the United States is expected to reach thirty-one billion dollars (Dodge 1994). The AIA predicts that ninety to ninety-five percent of the buildings that exist by the year 2000 have already been built (AIA 1988a).

Rehabilitation of existing structures makes good sense for a variety of reasons. At a very basic level, in a time of rapid changes, people and communities appear to need them as a physical reminder of their cultural and historical identity (Hoyt 1994, 86). As a result they are more likely to accept adaptive reuse over wholesale redevelopment. In terms of environmental concerns, buildings contain *embodied energy*, that is the energy required to construct them and produce the materials used in them. Demolishing the structure and building another to replace it loses this embodied energy and consumes more of the world's finite resources. Economically speaking, rehabilitation work can save up to eighteen percent over new construction costs because of a shorter construction time (ibid.).

According to statistics and public demand, the architectural profession is shifting from what the AIA 's Vision 2000 describes as an "Era of Building to the Era of Rebuilding," (AIA 1988b). This means that architects are going to have to learn how to deal with design issues and methods of construction that are different from the ones they use for new buildings. Ironically, new information technologies are potentially a way of dealing with old building information.

In the redevelopment process, design decision-making takes place in the context of a decision-maker applying specialized expertise to a particular building design instance. Computer images can now represent physical objects very accurately. Advances in digital design media have gone beyond simply the capabilities to produce two dimensional construction documents. Now there are opportunities to represent architectural building and contextual information in sophisticated two and three dimensional models on the architect's desktop. These types of representations are crucial for design techniques that rely on visualization. At the same time, advances in the field of artificial intelligence have brought forward not only an opportunity to represent design data in the computer, but expert knowledge as well. In recent years a new research has begun to create *Intelligent Computer Aided Design* [ICAD]† systems that can integrate all of these capabilities to assist the designer (Figure I.1.).

The purpose of this study is to determine if the integration of *digital heritage building records* and encoded domain expertise can support design decision-making for the redevelopment of historical structures. To accomplish this the study investigates the integration of a diverse set of information technologies: *Computer Aided Design* [CAD], *Knowledge Base Systems* [KBS], with a human Designer [Human Interface] in relationship to the redevelopment design process for heritage buildings. Together these technologies form a type of ICAD system called the *Smart Graphic Environment.* [SGE].

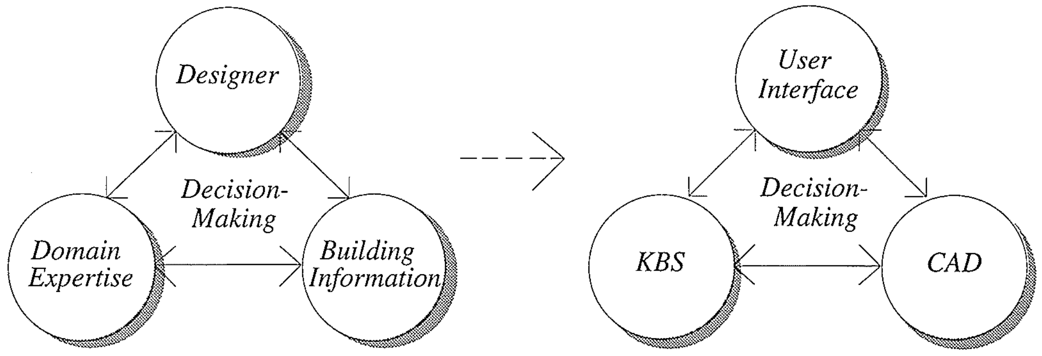


FIGURE I.1. - Computer Aided Design decision-making support systems

† The acronym ICAD was first used in reference to a research project undertaken at the school of Architecture and Environmental Design at the California Polytechnic State University (Pohl 1988). Recently an alternate acronym, KAD or Knowledge Assisted Design, has appeared and may, in time, become more widely used as a reference for these types of systems.

I.B. References

AIA, 1988a. "Vision 2000: trends shaping architecture's future," prepared for the American Institute of Architects by R.L. Olson and H. Kurent. Washington, D.C.: AIA Press, quoted in C. B. Kennedy, 1990. "Videographic documentation: analysis & manipulation." XXII *APT Bulletin.* Champaign IL: The Association for Preservation Technology Publication. 97-103.

AIA, 1988b. "Vision 2000: trends shaping architecture's future," prepared for the American Institute of Architects by R.L. Olson and H. Kurent. Washington, D.C.: AIA Press, quoted in N. Levinson. 1993. "Renovation Scoreboard." *Architectural Record.* (January) McGraw-Hill Publication. 69-73.

Dodge, F.W. 1994. "Rehab's projected growth." quoted in C.K. Hoyt, "More than preservation." 2 *Architectural Record.* (February) McGraw-Hill Publication. 86-87.

Hoyt, C.K. 1994. "More than preservation." 2 *Architectural Record.* McGraw-Hill Publication. 86-87.

Pohl, J.G. 1988. "ICADS - A prototype intelligent CAAD system." *ARCC Research Conference '88* . (proceedings) School of Architecture, University of Illinois, Urbana. 1-8.

Wotton, H. 1968. *The Elements of Architecture.* (facsimile reproduction of first edition London, 1624) Charlottesville: University Press of Virginia.

CHAPTER II

CONTEXT OF INVESTIGATION

This thesis explores the application of new computer technologies to the redevelopment design process for heritage buildings. The purpose of Chapter II is to begin this exploration with the identification of what the redevelopment design process is and the types of building information and domain expertise used to support decision-making. In particular, it is used to highlight the current status of research related to the use of digital technologies in the record and redevelopment of heritage buildings and the potential role of ICAD systems like the Smart Graphics Environment.

II.A. The Redevelopment of Heritage Buildings

The purpose of historical redevelopment is to integrate a new use in a heritage building without destroying historically significant features. Design problems presented by historical structures are more complex than those of a simple renovation. Not every historically significant building can be a museum. "Extending the life of a historical building requires recognizing its physical and socio-economic value as a historic structure while examining potential redevelopment that can support current and future usage" (Hill, et al. 1993). Although it is preferable to maintain the same use as a building was originally designed for, neighborhoods change over time, businesses become obsolete or relocate to less expensive sites, and building regulations become more strict. Rehabilitation assumes that at least some of an original building will have to be altered in some way to efficiently accommodate a contemporary use. However, this process cannot destroy historic character-defining spaces, materials, features, or finishes.

The main characteristic of the redevelopment process is that a substantial amount of time is spent at the beginning of the project collecting and analyzing existing building data. The redevelopment process can be summarized into three interrelated activities: (1) recording (2) analysis (3) application (or action) (Figure II.1). The purpose of recording is to acquire pertinent data and knowledge, whereas analysis provides understanding of this information. Understanding the building and recognition of important design issues are necessary before appropriate designs actions can be initiated.

Records are made both during and after the redevelopment process to document modifications made to the building. These records become an important frame of reference to measure change and for potential recovery of information when a significant building is damaged or lost. Unfortunately, many architects perceive extensive heritage recording as a luxury. In the interest of time saving or cost-cutting an architect may neglect this aspect of the design process, failing to realize the close relationship between a heritage record and the quality of decision-making. The availability of accurate and complete heritage building documentation impacts on the level of understanding of the structure, and therefore effects the ability of a designer to select appropriate conservation strategies, as well as opportunities for innovation, during the redevelopment process. Therefore, a computer system designed to assist decision-making in redevelopment projects must make it easy and cost-effective to access building information.

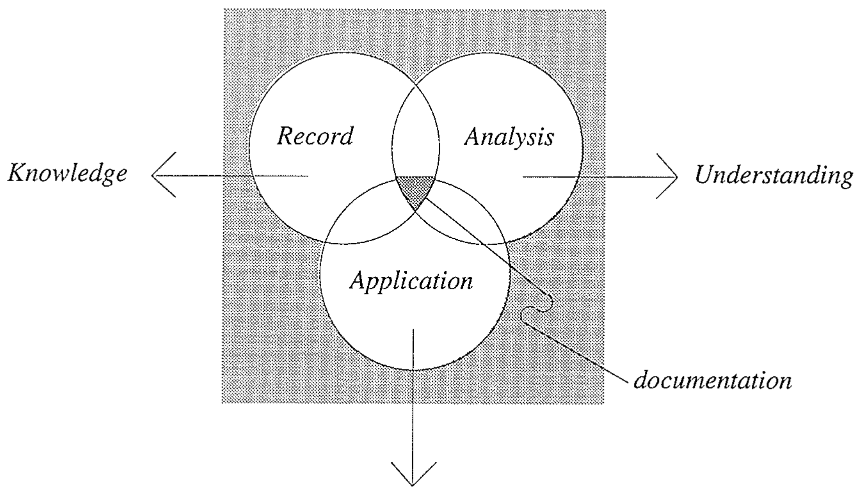


FIGURE II.1 - Inter-relationship of conservation activities (Stovel 1993)

II.B. Heritage Recording & Digital Records

Traditional heritage recording produces three basic forms of building documentation: (1) written, (2) photographic and (3) measured drawings. At the initial stages of design an informal photographic record of the building and context can be quite effective in communicating an idea of external impacts and identification of the project. For the purposes of project planning and design, and eventually construction, measured drawings containing extensive dimensions and annotations recording necessary historical and conditional information are required (Burns 1989, 120). Between manually collecting measurements from the building and converting it into drawing form, measured drawings take a long time to produce and it is for this reason that they are the most expensive form of documentation (Burns 1989, 110).

Documentation is used not only for design purposes and ongoing site maintenance and operation, but it also becomes an archival record that is used for historical study and as reference for the redevelopment of other structures. National and International agencies responsible for conserving and maintaining historically significant resources have developed high quality methods for recording and documenting heritage buildings.

In the past fifty-seven years over twenty-two thousand structures in the United States have been documented by Historic Buildings Survey and Historic American Engineering Record (HABS/HAER). This collection is growing at a rate of one thousand structures a year (Table 1). Parks Canada is responsible for keeping records for 120 National Parks, each with up to thirty historically significant structures, as well as all federally designated buildings, sites and structures. It is not hard to imagine the extraordinary volume of information contained within these heritage records. Not to mention finding ways to storing it, tracking it, and making it available to the public. The potential opportunities for the use of information technologies in historic conservation are extensive. However, the translation of heritage building data into electronic formats is in its infancy in North America.

TABLE 1 - HABS/HAER Records (Kapsch 1990)

Recorded Data	Per Year	to Jan. 1989
Bldg./Struct/Sites	1 000	22 590
Sheets of Measured Dwgs.	500	46 506
No. of Large Format Photos	5 000	127 177
No. of Data Pages	4 000	69 205
Photogrammetric Images	---	4 231

The level of utility of heritage records are established in terms of:

(1) accuracy,

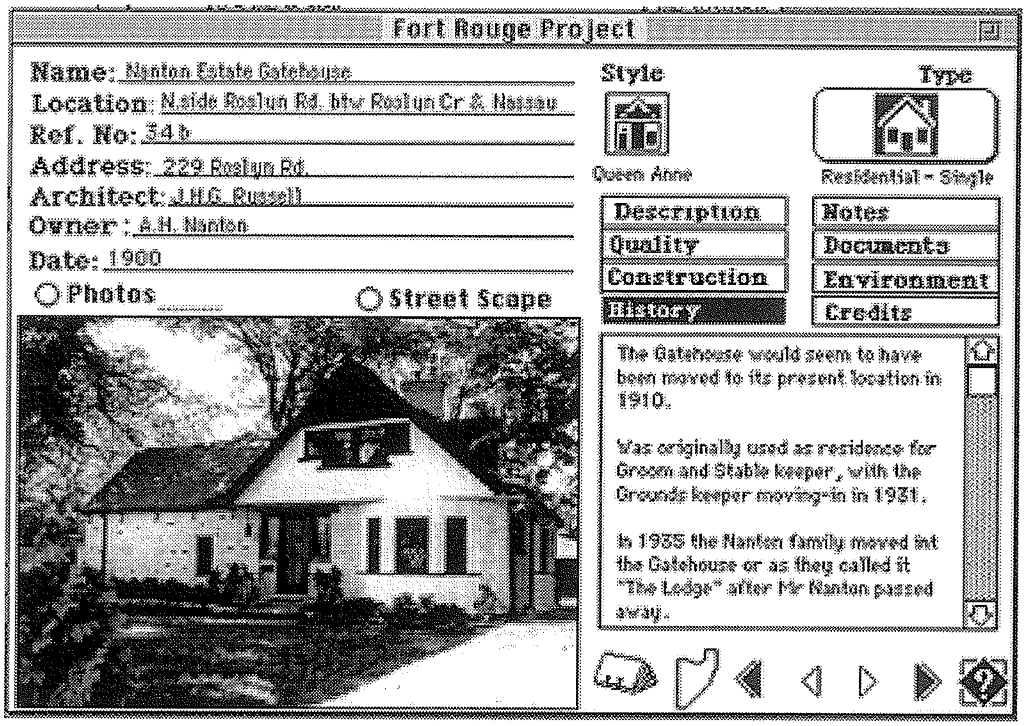
(2) retrievability,

(3) exchangeability, and

(4) legibility

(Stovel 1993). The quality of all of these characteristics is effected by the way information is recorded and stored. Although it is possible to achieve very high accuracy with some forms of traditional heritage recording such as hand measuring, the resulting paper and/or photographic documentation is neither compact nor portable. Organization and sharing of such documents is very difficult. The use of digital heritage records and recording techniques promises to address the problem of data under-utilization by providing a means by which the different types of heritage building data can be easily stored, manipulated and disseminated.

The first efforts in digitization of heritage resources were to create text- only databases that kept track of the location of building information, but not the information itself. For example in 1986, Parks Canada was using a database called Heritage Recording Management System (HRMS) that could identify the location of all archived heritage records. A similar system for HABS/HAER is described in (Kapsch 1990). More recently, research is being devoted to the exploration of the strengths, limitations, and advantages of using technologies that incorporate graphic capabilities such as *raster imaging*, CAD, *Geographic Information Systems* (GIS), and *hypermedia* (Figure II.2). However, a single standard approach to recording and documenting heritage buildings in a digital form has not yet been developed.



Name: Nanton Estate Gatehouse
Location: N.side Roslyn Rd. btw Roslyn Cr & Nassau
Ref. No: 34 b
Address: 229 Roslyn Rd.
Architect: J.H.G. Russell
Owner : A.H. Nanton
Date: 1900

Queen Anne

Residential - Single

The Gatehouse would seem to have been moved to its present location in 1910.

Was originally used as residence for Groom and Stable keeper, with the Grounds keeper moving-in in 1931.

In 1935 the Nanton family moved int the Gatehouse or as they called it "The Lodge" after Mr Nanton passed away.

FIGURE II.2 - Example of digital hypermedia file (with raster image) (Thompson 1993)

Standardization of digital heritage building data ensures the completeness of the data to communicate design concepts and allows the exchange of data between different computer programs and platforms. As part of architectural history heritage building information could potentially be disseminated to historians anywhere in the world, each with their own computer environment. The exchangeability of information between programs and environments is crucial if one intends to take advantage of the ability to reuse the same digital building data within the computer for a variety of purposes, from design analysis to inclusion in interpretive displays.

II.C. Heritage Building Design Analysis & Expertise

Historical redevelopment is a complex process because the designer needs to acquire the expertise from a variety of disciplines to analyze the building information.

At the preliminary stages of design there are a number of redevelopment issues to consider as one documents and examines the existing building conditions. Three principle issues are the: (1) legal, (2) physical, and (3) economic feasibility of the project (Shopsin 1989, 54). Within each of these broad categories there are a number of different specific analyses including:

(1) physical; i.e. identifying historical significance (classification),

(2) legal; i.e. establishing building code compliance (verification), and

(3) economic; i.e. integration of functional program (planning).

The ability to examine the building from these different perspectives requires expertise and skill in the areas of: architectural history, preservation standards and methods, building codes (as they apply to existing structures), building technology (including archaic materials/construction methodologies), older mechanical/electrical systems, facilities management, and so on. All of these issues are complex and interrelated. Inevitably in attempting to meet all of the different design goals conflicts arise. For example, how does one decide between repairing or replacing an nineteenth century window that doesn't meet today's building codes?

Having neither the time or the opportunity to acquire specialized skills, an architect is generally forced to rely on the services of outside professionals. Beside being costly, this is time-consuming and administratively complicated. This leads one to the question that if heritage building information can be recorded digitally and made available to a designer, then what about recording expert domain knowledge as well?

II.D. Expert Systems and Design Evaluation

Until relatively recently, computers with human-like knowledge, known as *Artificial Intelligence* (AI), were mostly considered as being in the domain of research laboratories and science fiction novels. That is, until the emergence of *knowledge base* or *expert systems.*†

Knowledge Base Systems [KBS] are able to store, organize, and manipulate large bodies of knowledge that are usually accessible to only a few experienced individuals. A KBS achieves high performance by using knowledge, as a human expert does, to make high quality decisions while minimizing the cost and risk involved. Although such systems may or may not replicate the same reasoning processes used by its human counterparts; the goal of the system is to produce the same results as a human expert would under the same circumstances.

Only a few prototype KBS have been developed for the specific purpose of evaluating heritage buildings. Those that have been developed do very specific tasks. Window Expert System [WES] is a prototype expert system used to diagnose and guide the repair and replacement of windows in historic buildings (Gilleard, et al. 1990). The prototype system developed by (Harada and Guckenheimer 1986) uses classification

† The term *expert system* is often used interchangeably with *knowledge base systems*. An expert system is a type of knowledge system with a much more limited scope within a single subject domain (Carrico 1989, 17).

techniques used by an architectural historian to identify to what architectural style or period a particular heritage building belongs. In order to solve problems both systems query the user about the building design. Like most KBS, neither of these prototype systems are able to retrieve building information directly from CAD or other digital graphic representations.

Over the past decade an ever-increasing number of researchers, including (Schmitt 1986), (Oxman and Gero 1987), (Coyne, et al. 1990), (Smasundaram and Mahabala 1991), (Rutherford 1992), (de Gelder and Lucardie 1993), have explored the development of different kinds of ICAD systems that are based on the idea of integrating CAD environments and expertise encoded in KBS.

II.E. Summary of Issues

In this chapter appropriate design decision-making in the redevelopment process for heritage buildings was described as being the result of analyzing carefully collected building information using special expertise. This process takes a substantial amount of time because of the need for the architect to deal with large amounts of different types of building data as well as scarce knowledge. The use of digital heritage building records was discussed as a means of increasing the utility of the building information. One of the problems cited about the utility of digital heritage building records was the current lack of standards for what these files would contain or how they would be formatted. KBS were introduced as a potential opportunity for providing designers with encoded expertise. ICAD systems, because they incorporate both design data and expertise in one environment, potentially create an ideal design decision-making environment.

The *Smart Graphics Environment,* described in the following chapters, is a particular type of ICAD that focuses on digital drawing information of heritage buildings. The SGE is conceived as an interactive design decision support environment. Unlike other retrieval or query-based expert systems currently developed for use in heritage building redevelopment, the designer would be able to initiate critical analysis of digital heritage building files directly from within a CAD environment. This thesis, therefore, includes approaches for the different stages of ICAD system development including: (1) data identification and collection, (2) representation and formalization, (3) component integration, and (4) implementation and testing.

II.F. References

Burns, J.A. (ed.) 1989. *Recording Historic Structures.* National Park Service. Washington, D.C.: American Institute of Architects Press.

Coyne, R., M. Rosenman , A. Radford, M. Balachandran, J. Gero. 1990. *Knowledge-Based Design Systems.* Sydney: Addison-Wesley Publishing Company.

de Gelder, J. and G. Lucardie. 1993. "Knowledge and data modeling in CAD/CAM applications." *Design & Decision Support Systems in Architecture.* H. Timmermans (ed.) The Netherlands: Kluwer Academic Publishers. 111-122.

Gilleard, J., J. Myers, A. Olugbemiga. 1990. "Computer applications in architectural conservation." *Acadia '90 Proceedings - From Research to Practice.* P.J. Jordan (ed.) Hawaii: ACADIA. 187-199.

Harada, L. and S. Guckenheimer. 1986. "An expert system for classifying nineteenth century American residential architecture." (unpublished paper) Winter Term. Boston: Harvard University.

Hill, S., M. Evans, L. Strachan, P.R. Perron. 1993. "The use of smart graphics for the redevelopment of historical structures." presented at *Design & Decision Support Systems Conference,* 6 - 10 July 1992. Eindhoven, The Netherlands. [forthcoming in the Journal of Architectural Research and Planning].

Kapsch, R.J. 1990. "HABS/HAER: A User's Guide." XXII. *APT Bulletin.* Association for Preservation Technology. 22-34.

Oxman, R. and J.S.Gero 1987. "Using an expert system for design diagnosis and design synthesis." *Expert Systems.* 4 No. 1. 4-15.

Rutherford, J. 1992. "Knowledge-based design decision support." presented at *Design & Decision Support Systems Conference.* 6 - 10 July 1992. Eindhoven, The Netherlands.

Shopsin, W.S. 1989. *Restoring Old Buildings for Contemporary Uses.* New York: Whitney Library of Design.

Somasundaram, M. and H. Mahabala. 1991. "A knowledge based critiquing approach to floor-plan evaluation." *4th UNB AI Symposium* . 19 - 21 September 1991. Fredericton NB, Canada. 483 - 293.

Stovel, H. 1993. "An introduction to heritage recording principles and practices." (Training Course Lecture) presented at *New Low-Cost Computer Tools for Heritage Conservation Practices & Built Environment Sustainable Development Programs Course* . 15 November 1993. Ottawa: Government Services Canada.

Schmitt, G. 1986. "Expert systems in design abstraction and evaluation." *The Computability of Design - 1986 Suny Buffalo Symposium on CAD.* A.C. Harfman, Y.E. Kalay , B.R. Majkowski, L.M. Swerdloff (eds.) Buffalo: State University of NY Press. Part 2.

Thompson, W.P. 1993. "Historical buildings of manitoba." (Hypercard™ stack program.) Winnipeg, MB: Faculty of Architecture, University of Manitoba.

CHAPTER III

THE INVESTIGATION

III.A. Research Objectives

The goal of this research was to prove that the integration of digital heritage records and KBS can provide design decision-making support for the redevelopment of heritage structures. The objectives of the research were to:

(1) investigate and define a conceptual framework for creating and integrating the components of a Smart Graphic Environment. The components include:

   i) Knowledge Base System

   ii) Digital Heritage Building Record (CAD and Integrated Database)

   iii) User Interface (Human Designer)

(2) implement and test concepts in a working prototype.

This study concentrates on heritage buildings that are potentially suitable for adaptive re-use. Structures under consideration are limited to those whose historical significance is established by physical or visual features, as opposed to association with important events or persons. Such buildings have distinctive physical characteristics of a type, period, or method of construction and therefore lend themselves well to graphical representation and analysis using CAD.

III.B. Overview of Investigation

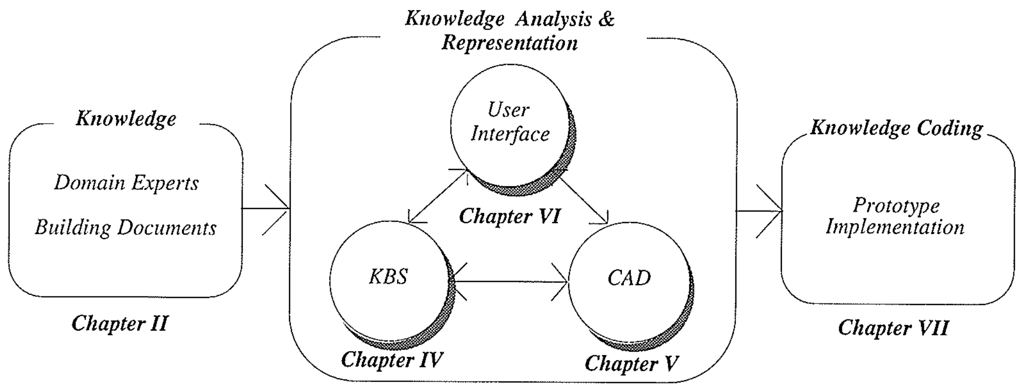


FIGURE III.1 - Overview of investigation

The following Chapters IV through VI describe a conceptual framework for the development of a Smart Graphic Environment. The purpose of the framework is to initially establish, at a conceptual level independent of any particular implementation scheme, how each of the components can be created on a computer and then linked together. The components of the SGE are addressed separately according to the stages of KBS system development based on (Carrico et al. 1989) (Figure III.2). In this study, the KBS system component is investigated first to determine what design analysis tasks such a system is capable of accomplishing and its related data requirements.

The conceptual framework developed in this study is tested through the implementation of an actual prototype system in commercially available software and hardware that might be found in an average architectural office (described in Chapter VII). This aspect of the research was greatly facilitated by consultation with an expert system developer[†], who was able to provide detailed technical and programming support required to create the prototype.

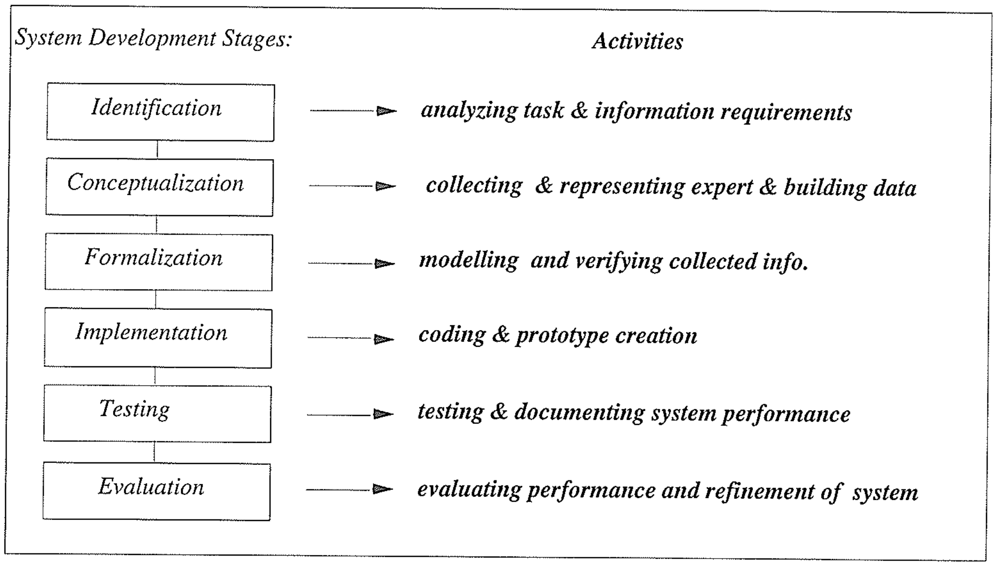

System Development Stages:
Activities
Identification
analyzing task & information requirements
Conceptualization
collecting & representing expert & building data
Formalization
modelling and verifying collected info.
Implementation
coding & prototype creation
Testing
testing & documenting system performance
Evaluation
evaluating performance and refinement of system

FIGURE III.2 - SGE System development stages & activities

III.C. System Development - Object-based Approach

CAD systems, KBS, and human designers deal with complex information in different ways. To achieve expert evaluation, all of the different components of a Smart Graphics Environment have to exchange information, work together, and interact *intelligently* (after Manola, et al. 1992).

† Dr. Mark Evans, Department of Computer Science, University of MB.

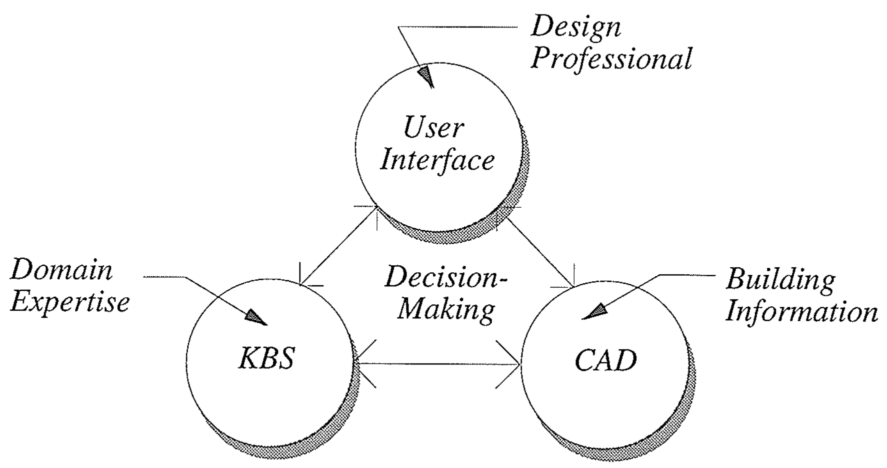


FIGURE III.3 - Conceptual framework of SGE components

In order for the computer to understand drawing information in the same way as a human designer, information implicit in a drawing must be made explicit or the knowledge for doing this must be provided (Coyne, et al. 1990, 94). At the same time, the design knowledge required to evaluate the building design also has to be encoded in a knowledge base. This explication process is often described as being analogous to developing a *language of design*. Like the words of a language, a design is made up of component parts that are taken from a broader vocabulary of parts. Parts are assembled together according to certain rules or "grammars" (design knowledge) that describe the relationship between parts.

An efficient approach for representing both design data and design knowledge is based upon object-oriented computer programming. Object-oriented development techniques attempt to manage the complexity inherent in real world tasks by abstracting out knowledge and encapsulating it within objects. Each object† represents a unit of

† Objects are not necessarily only physical entities, but can be phenomena or processes also.

information (such as a "door"). Objects can have attributes or properties, relationships with other objects, and belong to *object classes* with characteristics common among groups of objects (for example, a door belongs to the class "openings," and therefore shares the characteristic of "connects two spaces together" with other members of its class, such as "window"). An *object instance* describes a specific object, such as a "principle entrance door." The object-oriented approach encourages a view of the world as a system of cooperating and collaborating entities. Work is accomplished by manipulating the objects -- creating objects, modifying objects, invoking functions associated with the objects, etc. The object-oriented approach offers numerous advantages such as: modularity, reusable components, encapsulation and abstraction; all of which help alleviate the complexity involved in developing and maintaining large applications (Coad and Yourdon 1991).

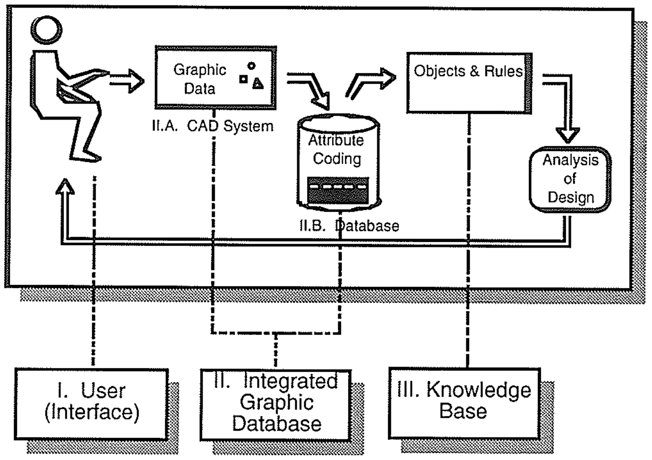


FIGURE III.4 - The Smart Graphics Environment

Based on a shared object-oriented description of design data and design knowledge it should then be possible to create a Smart Graphic Environment with components that are able to exchange, store, manipulate information, and perform analysis tasks to assist the designer. In addition to user interface, these components include an object-based CAD and *Integrated Graphic Database* system [IGDB], and an object-based KBS (Figure III.4).

III.D. References

Carrico, C., J. Girard, J. Jones. 1989. *Building Knowledge Systems.* New York: McGraw-Hill Book Co.

Coad, P. and E. Yourdon. 1991. *Object Oriented Design.* Englewood Cliffs: Yourdon Press.

Coyne, R., M. Rosenman, A. Radford, M. Balachandran, J. Gero. 1990. *Knowledge-Based Design Systems.* Sydney: Addison-Wesley Publishing Company.

Manola, F., S. Heiler, D. Georgakopoulos , M. Hornick, M. Brodi. 1992. "Distributed object management." *International Journal of Intelligent and Cooperative Information Systems.* 1(1) Waltham, MA: World Scientific Publishing Company. 5 - 42.

CHAPTER IV

KNOWLEDGE BASE SYSTEMS AND REDEVELOPMENT DESIGN

The purpose of the following chapter is to describe the development of a KBS component for a Smart Graphics Environment. This chapter begins with an overview of the design process and then highlights the potential role for KBS in providing design decision support by analyzing digital heritage building records. Problems related to creating a KBS are outlined including: identifying, conceptualizing, formalization, and sharing of design knowledge and data. The benefits of using an object-based approach to overcome these problems in capturing design expertise in a KBS are presented.

IV.A. The Design Process

The development of an intelligent design decision support system requires careful attention to the nature of the design process itself. The earliest practical expert systems, like the medical diagnosis program *MYCIN*, had very narrow and self-contained domains of knowledge with very clear goals and strategies for reaching those goals. The most successful intelligent design support systems have been ones that help designers solve very clearly defined problems. For the most part, building design is a largely unstructured activity where goals and strategies are not predefined but evolve through the process itself (Drach, et al. 1993, 64). Design knowledge and data are in fact quite dynamic, where attributes and relationships between attributes can change depending on the context in which they occur (de Gelder and Lucardie 1993, 111). This is why special techniques are required for acquiring and representing design knowledge and data.

Design is an iterative process where architects use their expertise to approach a set of ideal solutions from a larger number of possibilities. The three-phase design model originally introduced in (Asimow 1962) and reiterated by (Coyne, et al. 1990) is broadly accepted as a general framework used by designers. The phases include: *analysis, synthesis,* and *evaluation* (Figure IV.1.).

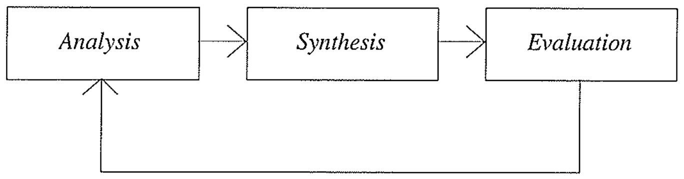


FIGURE IV.1. - Design phases

> The first task is to diagnose, define, and prepare -- that is to understand the problem and produce an explicit statement of goals. The second task involves finding plausible solutions. The third task concerns judging the validity of solutions relative to the goals and selecting among alternatives. A cycle is implied in which the solution is revised and improved by reexamining the analysis. (Coyne et al. 1990, 12)

The role of the KBS component in the Smart Graphic Environment is intended to provide design decision support specifically for the **redevelopment** design process. For the purposes of this study emphasis is placed on the analysis phase of the design process. Analysis is a particularly critical phase of the redevelopment design process, since it is at this stage that existing condition of the building is assessed, design problems are defined and project goals are determined.

<u>IV.B. The Role of Analysis in Redevelopment</u>

In the case of heritage building redevelopment, the design process can be considered as one of model refinement. This type of design process is classified as being parametric or *routine design* (Schmitt 1990) because the design problems are well-defined. A designer begins with a number of known parameters of an existing structure and those of a proposed new use. In general, because they are "historic", the basic designs of such buildings cannot be fundamentally changed. Therefore, the potential number of solutions is limited. Through analysis, parameters of the design are clearly defined and thus the designer is brought closer to potential solutions. In theory then, solutions are merely the result of adjusting parameters to suit the specifications of design goals. However, in practice this process is not as routine as this theory suggests.

In redevelopment there are often direct conflicts between design goals. Achieving modern convenience and safety requirements while preserving features not originally designed for this purpose can be a difficult challenge. It is for this reason that this type of design work is often characterized as being one of adaptation and compromise as well as involving some degree of innovation.

The role of expert analysis in redevelopment design process is extremely important. Before appropriate treatments or renovations can be decided upon issues such as: building code compliance, the ability to support new use(s), and the need to maintain historical integrity, all need to be clearly identified and understood. Not meeting building codes can severely compromise the life and health safety of a building's occupants. Certain plan configurations or original uses may easily accommodate certain types of new use. In other buildings plan configurations may not be suitable, resulting in a negative impact on the economic feasibility of the project. A designer who is unfamiliar with different architectural styles may inadvertently remove or alter significant aspects of the building. Those unfamiliar with period construction techniques may choose inappropriate treatments or repairs. Gaining understanding and identifying key design goals and objectives are valuable products of the analysis process. Expertise is used to solve problems by employing strategies for identifying what to look for and how to break complex problems into more manageable parts.

IV.C. Developing a Knowledge Base Design Decision-making Support System

Architects working in historical redevelopment encounter a number of design challenges that require expertise that is not usually common in an architectural practice unless it is committed to historical building conservation. Taking the time to acquire knowledge by extracting it from domain literature, attending training courses and/or hiring an external consultant(s) are very costly options. In this context, providing expert analysis on the architect's desktop using computerized expertise is a valuable alternative.

KBS use expert knowledge in a domain and information about objects and relationships described within that domain of discourse to reason about a situation in order to solve a problem or give advice (Carrico 1989, 17). The ability to use *heuristics* or "guessing" strategies in the same way human experts do to solve problems is what sets such systems apart from other search and retrieval programs and is the hallmark of a true KBS.

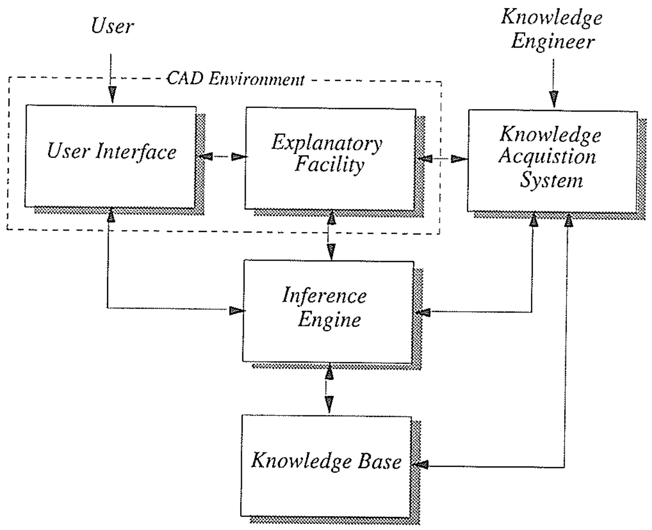


FIGURE IV.2. - Components of an expert system (after Ballast 1991)

A KBS contains two distinct components: the *knowledge base* and a control mechanism (strategies) for reasoning about the knowledge called the *inference engine* . The knowledge base contains a representation of expert knowledge, usually in the form of facts (problem-specific-data) and decision rules. Problem-specific data is stored and accessed from a database (sometimes referred to as *working memory*). Each knowledge base is specific to a particular problem domain. In addition to the knowledge base and inference engine, KBS also have an *user interface*, *explanation facility* (describes how conclusions are reached by the system), and a *knowledge acquisition facility* (used to enter and modify knowledge in the knowledge base (Figure IV.2.).†

The most difficult aspect of creating a KBS is the process of translating facts and human expertise into a language the computer can understand. This process is usually undertaken by a Knowledge Engineer. Knowledge engineering involves:

† In the SGE user interface and the explanation facility are provided through the CAD environment (see also Chapter VI - The Designer).

(1) defining what the system will do,

(2) finding and collecting the expertise needed,

(3) representing and verifying expertise, and

(4) translating expertise into a machine-usable form.

Once this expertise is encoded on the computer the KBS has to be supplied with (5) the relevant information about a given problem, and, be provided with a means of giving feedback and analysis through a suitable (6) user interface.

Two challenges that are unique to the KBS component of the Smart Graphic Environment are the requirements to: (1) represent and integrate **design expertise** on the computer, and (2) to access heritage building design information directly from a digital (CAD) model.

IV.C.1. Analysis paradigms - determining what the system will do.

Experts from different domains will look at redevelopment design problems in different ways. The first task in identifying what expert knowledge needs to included in a KBS is to establish a *paradigm.* Paradigms (patterns or models) are used by KBS developers to describe how a given expert *thinks* about a problem within a given domain (Carrico, et al. 1989, 27). Knowing this helps in structuring and eventually representing knowledge.

Two typical analyses in the redevelopment of heritage buildings are: (1) the identification of the style or period to which a building belongs, and (2) establishing the building code compliance of an existing structure. These analyses represent two different KBS paradigms -- classification and verification.

Classification is when knowledge in a KBS is used to select the best solution to a problem from a variety of alternatives. The prototype expert system developed by (Harada and Guckenheimer 1986) used to identify nineteenth century residential architecture (employing the same classification techniques followed by an architectural historian) follows this paradigm.

In the case of a verification paradigm, knowledge is used to identify cause and effect relationships from amongst numerous possibilities (Figure IV.3.). For example, Window Expert System [WES] is a prototype expert system used to diagnose and guide the repair and replacement of windows in historic buildings (Gilleard et al. 1990). It uses an expert system to narrow down potential repair actions depending on the significance and condition assessments of windows. Similarly, building code compliance can be determined from a comparison and classification of building performance against a number of design requirements.

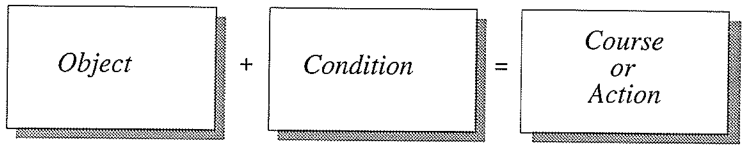


FIGURE IV.3. - Verification

Within the context of this study only the verification paradigm (for the purpose of establishing code compliance) was explored in detail and implemented (Chapter VII). Although it would be desirable to include more than one type of analysis in the system it soon becomes obvious that such a system would have to incorporate additional expertise or *meta-knowledge* for resolving conflicts that arise between conclusions of different electronic experts.

The key behind the KBS systems, like the two prototype expert systems cited above, are that they do very specific tasks. This allows paradigms to describe fairly accurately how an expert may deal with choosing the best solution to a specific design question like how to fix a window or identify what style the building is. A clearly defined task makes it easier to identify all of the procedures used by the domain expert. The computer is, after all, a novice capable of doing only what it is instructed to do. A large part of knowledge engineering relies on the domain expert being able to clearly describe what it is that they do to solve problems. A task human experts often find very hard to do.

IV.C.2. Knowledge and design - acquiring special expertise. There are two basic sources of expertise for a KBS: (1) domain documents and (2) the experts themselves. Domain documents are used to establish facts and rules (goals, parameters, definitions) and experts can explain how best to use and apply them.

The best technique for gathering knowledge is to interview domain experts and observe them at work (Carrico, et al. 1989, 43). In addition to understanding the fundamental knowledge of their subject domains, experts have established techniques for reasoning that are more efficient than techniques a novice would use. Heuristic type knowledge is self-taught over the years through exploration and experimentation. So, for example, if one wishes to learn how a building official examines a heritage building records for building code conflicts, the best approach is to give them several examples and ask them to describe what problems they are looking for and how they are looking for them. The goal is to establish a standard set of conventions, or a *protocol,* for how data is treated. Protocols for design evaluation can also be established by analyzing domain documents, such as building codes and books on historical building styles. Using the same protocol, a novice should be able to get the same results as the expert. If not, then the protocol is invalid and the procedures need to be reviewed and clarified.

One of the important outcomes of the acquisition of expertise is the identification of knowledge base objects that are used in a particular analysis. This helps to model and encode knowledge within the computer as well as determine the data requirements for heritage building files.

Experts and domain documents use numerous implicit and explicit object definitions and requirements. The nature of these objects can be quite complex and varied even within a single domain of discourse, for example some building code objects are described by what they are ("door") or by what they do ("firewall"). The rules and definitions for objects found building code documents are generally more performance-based than prescriptive. For example, instead of saying all windows must be a certain type, a rule may describe an acceptable level of heat-loss/gain permitted. By being purposefully generic the same rules can be applied to a variety of design instances. At the same time, it makes it difficult to define objects for computer it can identify within given digital geometric CAD building model.

Understanding the connection between design knowledge and design data is crucial when both are to be integrated within one environment on the computer. The prototype expert systems by (Gilleard, et al. 1990) and (Harada and Guckenheimer 1986), described earlier in this section, get heritage building data, not from a CAD drawing, but from the designer. The expert system does this by asking pointed questions. Potential answers to these questions are limited and predictable (yes/no, or a number within a certain range). The responses from the designer are used by the system to draw conclusions within its particular area of expertise. In such query-based environments the expert system tells the designer what information it needs and when it needs it. The significant difference between the Smart Graphics Environment and many other KBS design systems is the fact that much of the information it gets about the design is from the drawing itself and not by querying the user. When the expert system draws its information from a CAD drawing file the designer has to anticipate what information it will need and include it in the file.

Given a single construction industry standard for how information is to be defined it should be relatively simple to create a drawing and database file that contains all of the design information needed for any architectural and engineering analysis. This is what is known as a *product* or *building* (data) *model.* It contains of all of the objects and object attributes that make up a building design. Despite ongoing efforts to create a standard for data definitions for building objects and object attributes, particularly for CAD/CAM applications, universal agreement within the construction industry has not been reached yet. The primary stumbling block is that architectural objects and their descriptions change depending on the context of the analysis. The same design entities can be described in very divergent ways. An architect may look at a wall as part of the organizational system of the building, analyzing the pattern of openings or division of space. A structural engineer may be concerned a wall as part of a structural system and analyze it in terms of wind resistance, or tension and compression.

The definition of building design data comes from the domain of the expert. More specifically it arises from the data requirements of the analysis procedure itself. A prevailing characteristic of rules used to analyze building designs is that they can accommodate a number of different design solutions. This means that design analysis is occurring not at the physical level of each building component, but at a conceptual level involving the functions and relations that these objects embody about a design. Considering domain knowledge from this perspective supports the idea of objects being different but still sharing a functional equivalence.

<u>IV.C.3. Conceptual Models.</u> Examining a subject domain using a knowledge level analysis allows one to study a domain's *conceptual* (data) *models* before trying to implement them in a computer environment (de Gelder & Lucardie 1993, 111). Conceptual models represent domain knowledge and data in terms of *concepts*, sometimes considered as *goal-states*. They define a solution space and the interdependencies involved. Conceptual models explain design constraints in terms of the classes and hierarchical relationships at a cognitive level that is more common to the way designers work.

Conceptual models describe *design elements* (objects) and their attributes in terms of function or relationship to one another. Although they can be used to describe physical (R-value) and non-physical (color) attributes of objects, they are generally not used for geometric or shape descriptions (that are more readily described using CAD representations). Design elements can be spaces (rooms, corridors), building components (roof, door), and/or systems (air-conditioning, plumbing). *Associations* (also called *relations*) are the relationships between design elements (is a, part of, type of).

The best way to organize and validate conceptual models is to draw them. This is done using a conceptual graph, or *semantic network* (Figure IV.4.). In these diagrams concepts (nodes) are connected together with labeled lines of associations. Associations between objects change depending on the what aspect of design knowledge is being represented (a structural concept is different from a functional one). Often associations are interrelated so that management of complex models can become problematic (Drach, et al. 1993, 71). The Nijssens Information Analysis Method [NIAM] is an example of a sophisticated formalized technique for modeling many different conceptual relationships (see also Chapter VII). *Type hierarchy* diagrams are also used to show how objects inherit the attributes of another (Figure IV.5.).

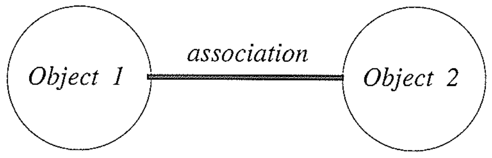


FIGURE IV.4. - Semantic network

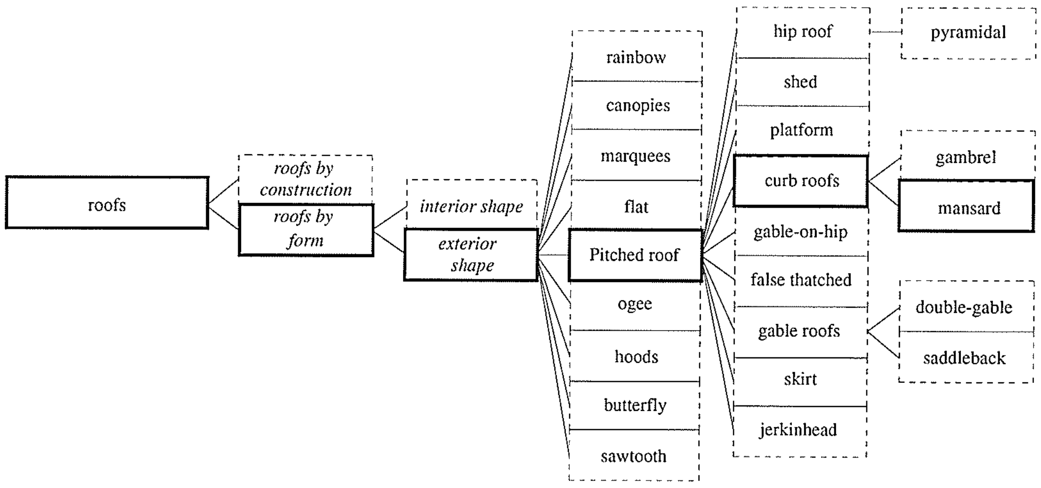
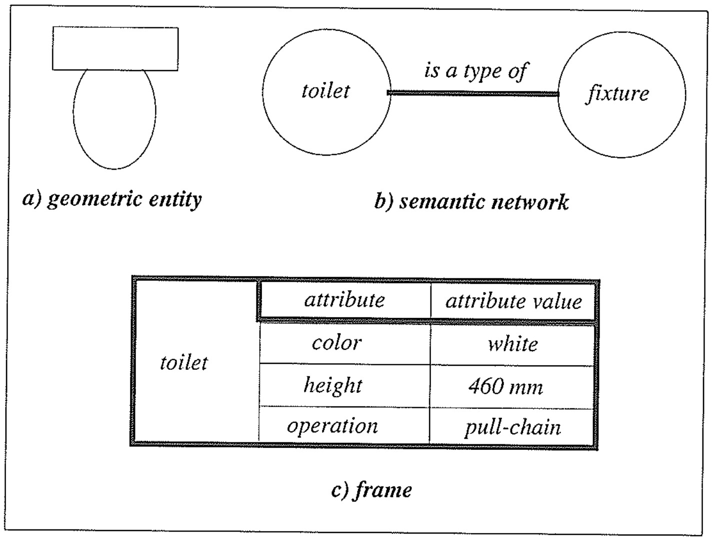


FIGURE IV.5 - Type hierarchy for roof object
(based on *Art & Architecture Thesaurus* 1990)

One of the advantages to conceptual modeling is that domain experts find it easy to understand, particularly when compared to reading computer program code. Thus, conceptual models allow the expert to verify the objects and relationships contained in the model. It is interesting to note that the explication of domain expertise often provides insights about the analysis process that even the experts themselves do not realize. In recent years, the conceptual modeling of building codes is helping to identify oversights and inconsistencies in building code documents.

Clarity of design intentions and goals is particularly advantageous when one has to resolve conflicts between them (Drach, et al. 1993, 68). Conceptual modeling makes such conflicts explicit so that they can be addressed, either by the human designer or eventually with a computer system with embedded meta-knowledge that specializes in solving such conflicts .

Conceptual modeling provides a technique for formalizing and making explicit the internal model human experts use to think about design problems. A domain expert reasons by associating their own design knowledge with design data. In this way experts recognize problems, critical relationships, and taxonomies that are a part of achieving design goals.



toilet	attribute	attribute value
	color	white
	height	460 mm
	operation	pull-chain

FIGURE IV.6. - Example of three object descriptions:
a) CAD, b) semantic network, and c) object frame (relational)

Chapter VII provides further descriptions and examples of the use of conceptual modeling in the context of implementation of a prototype system that included representing knowledge and data for determining the building code compliance of heritage buildings.

Figure IV.6. illustrates an three different types of object descriptions used in the prototype system.

IV.D. Object-based Approach for KBS

Conceptual models help define and clarify design knowledge and data requirements used in the analysis of heritage buildings. The next stage of development of the KBS component of the Smart Graphic environment is to translate these models into a machine-usable form.

The most common type of knowledge representation in a knowledge base is with rules. Each rule is a simple conditional statement [IF] consisting of one or more preconditions and a [THEN] conclusion. If the preconditions are satisfied then the associated conclusion is drawn:

```
IF....THEN....              IF (object=Window)
                            THEN (object class=opening)

IF....AND.....THEN....      IF (object=Window)
                            AND (operation=swing out)
                            THEN (object type=casement)
```

Rules work particularly well when a goal can be met by following a logical sequence of steps. In the example above the system is able to identify all windows that swing out as being part of a class of "opening" called "casement." In such an environment data definitions remain static; a window does not become a "mirror" or an "thermal barrier", unless more rules are added to describe these conditions also.

Tasks that can be easily represented by a *decision tree* generally can be represented in a rule-based system. The "limbs" of a decision tree take you to a node whereupon each decision directs you onto a path that reduces the number of conclusions until there is only one answer (the "leaf"). A KBS that determines the style of a building is systematically narrowing down potential styles based on the certain character-defining features of each style and asking if they are present on the building being examined. Conceptual models, unlike decision trees, do not represent a procedural approach at all. They tend to concentrate more on the defining a solution space on the basis of functions and relations between design objects.

The representation scheme for the Smart Graphics Environment combines a rule-based approach with an object-based approach. In this hybrid approach, data in the knowledge base is grouped into objects. Rules and procedures, that can manipulate the data, are located directly within each object.

The Knowledge Base is organized as a set of object classes that represent templates or models of functional constraints, object characteristics, and contextual or object-to-object relationships that are common to groups of objects (Figure IV.7. and VII.11.). A set of procedures and rules is encapsulated in each object to specify this knowledge and how it should be applied in evaluating specific object instances (these object instances are retrieved from an Integrated Graphic Database connected to a CAD building file). Classes are arranged hierarchically so that they can share information. Subclasses inherit the characteristics of their parent classes (for example, a washroom door will inherit the shared characteristics of all doors). Standard defaults are used to enable evaluations when information about the existing building and its component objects is limited or incomplete. This is a powerful mechanism for dealing with partial information (i.e., prototypical information). Defaults can be overridden when actual values for an object instance's attributes become known (Hill, et al. 1992).

Object	=	{slot1, slot2, ... slotN}	
Slot	=	{facet1, facet2, ... facetM}	
		a slot contains information associated with an object's attribute	
Facet	=	Value	*actual value or default for classes*
		Possible-Values,	*description of possible values*
		Legal-Values,	*constraints representing legal values*
		Explanation,	*justification for constraints*
		Value-Type,	*numeric, list, string, etc.*
		When-Changed,	
		When-Needed,	
		When-Removed	

Note: the When-? facets are used to process events that correspond to the facet name. For example, the When-Changed facet provides instructions to be carried out when the value in the slot of an object instance is changed. This may involve updating other slots in the object instance or creating or deleting other objects.

FIGURE IV.7 - Object-based representation (Hill, et al. 1992)

The benefit of using an object-based approach is that rules and constraints are encoded in appropriate object classes in the knowledge base. The rules state the required characteristics of the object classes to ensure that design goals, such as compliance with building regulations and historical preservation guidelines, are met. CAD drawings provide a set of appropriate object instances (*As-Built or As-Designed* state). The object instances are examined by the expert system for potential violations with design goals (*As-Required* state). Violations are reported to the user and methods for complying with the violated goals are presented. These methods must be defined in the object classes in the knowledge base so that they can be applied to specific object instances when violations arise.

The expert system contains an inference engine that interprets and manipulates object classes and instances to perform design evaluations. The basic functions of the SGE inference engine are to :

- Extract object instances from the Integrated Graphics Database and create appropriate object instances in the knowledge base.
- Manipulate these object instances and their associated object class knowledge to enforce constraints, explain violations, suggest methods for resolving violations, etc.

In an object-based knowledge base, control is typically distributed throughout the object classes and subclasses defined in the knowledge base. This is done by associating special-purpose procedures (called *methods*) with one or more object classes. Methods interpret and manipulate the contents of the objects that they are contained in. Like data, methods can be inherited from parent classes. This allows general methods for interpreting objects to be shared among many object classes (and hence their subclasses and instances). It also allows specialized methods to be represented directly with objects when necessary. A general control mechanism coordinates the selection of the methods that should be used (it handles inheritance and conflict resolution when necessary). The advantage of this scheme is that the methods for manipulating objects are defined within the objects rather than in a large body of code as is the common approach in procedural computer programming languages such as Pascal and C. Linking the methods of manipulating an object with the object itself makes it much easier to maintain a large knowledge base (Hill, et al. 1993).

IV.E. References

Art and Architecture Thesaurus. 1990. Vols. I - III. J. Paul Getty Trust. Oxford: Oxford University Press Inc.

Asimow, M. 1962. *Introduction to Design.* New Jersey: Prentice Hall Inc.

Ballast, D. 1991. "New kid on the block." *Architectural Record*. Vol. 179. February. 40-41.

Carrico, C., J. Girard, J. Jones. 1989. *Building Knowledge Systems.* New York: McGraw-Hill Book Co.

Coyne, R. 1990. "Logic of design actions." *Knowledge Based Systems.* 3 No. 4. Sydney: Butterworth-Heinemann Ltd. 242-257.

de Gelder, J.and G. Lucardie. 1993. "Knowledge and data modeling in CAD/CAM applications." in *Design & Decision Support Systems in Architecture.* Harry Timmermans (ed.). Netherlands: Kluwer Academic Publishers. 111-122.

Drach, A, M. Landenegger, S. Heitz. 1993. "Working with prototypes, from CAD to flexible tools for integrated building design." in *Design & Decision Support Systems in Architecture.* H. Timmermans (ed.). Netherlands: Kluwer Academic Publishers. 61-82.

Gilleard, J., J. Myers, A. Olugbemiga. 1990. "Computer applications in architectural conservation." *Acadia '90 Proceedings - From Research to Practice.* P.J. Jordan (ed.) Hawaii: ACADIA. 187-199.

Harada, L. and S. Guckenheimer. 1986. "An expert system for classifying nineteenth century American residential architecture." (unpublished paper) Boston: Harvard University.

Hill, S., M. Evans, L. Strachan. 1992. "Smart Graphics -- An approach for automated Expert Evaluation of historical structures." Presented at *ICOMOS Canada Computers in Conservation Conference*. 16 - 18 Aug. 1992. Quebec City, Canada. [forthcoming in APT *Bulletin*, the Journal of the Association for Preservation Technology].

Hill, S., M. Evans, L. Strachan, P.R. Perron. 1993. "The use of smart graphics for the redevelopment of historical structures." presented at *Design & Decision Support Systems Conference.* 6 - 10 July 1992. Eindhoven, The Netherlands. [Pending publication in the Journal of Architectural Research and Planning].

Schmitt, G. 1986. "Expert systems in design abstraction and Evaluation." *The Computability of Design - 1986 Suny Buffalo Symposium on CAD.* A.C. Harfman, Y.E. Kalay, B.R. Majkowski and L.M. Swerdloff (eds.) Buffalo: State University of NY Press. Part 2.

Schmitt, G. 1990. "Classes of design - classes of tools." *Electronic Design Studio.* M. McCullough, W.J. Mitchell and P.Purcell (eds.). Cambridge: MIT Press. 70 - 90.

CHAPTER V

DIGITAL HERITAGE BUILDING FILES

The computer's ability to store, manipulate, retrieve and disseminate heritage building information is far superior to traditional documentation techniques. The most important advantage to creating digital heritage building files is a potential to increase the utility of digital data so that it can be shared by a variety of programs within the computer. The role of the CAD component of the Smart Graphic Environment is to provide access to heritage building design data to the user and KBS component for analysis. The purpose of this chapter is to (1) outline the types of data that are traditionally found in a heritage building record, (2) techniques used to collect data and convert it into digital graphic formats, and finally, (3) to describe standards for creating an attributed object-based CAD and *Integrated Graphic Database* [IGDB] digital heritage building file usable by a KBS.

V.A. Extending the Utility of Heritage Building Records

Traditional heritage recording produces three different categories of building documentation:

(1) Unscaled Graphic Information (UGI) - i.e. photographs

(2) Scaled Graphic Information (SGI) - i.e. measured drawings

(3) Text & Tabular Information (TTI) - i.e. written material, annotations

The primary source of this information is obviously the heritage building itself, but additional information can be found in original plans, historical photography, paintings and sketches, and archival records (McKee 1970).

Although they are used to represent actual physical objects, photographs and architectural drawings are abstract in nature. This means that most of the valuable information that they contain is interpreted by an observer based on previously acquired knowledge and experience. Different information can be extracted from the same representation depending on the expertise of the observer.

Visual and geometric building representations are often the sole source of information used to analyze a number of important aspects of a redevelopment project. Architectural historians can determine the period and style of a building from observing visual features of the exterior of a building. Building authorities can identify many aspects of code compliance by examining plans. Planners and facility managers also use site and building plans to assess spatial efficiencies and property values. Building engineers use photographs to examine evidence of common problems as well as identify building materials. Presented with a three-dimensional structural model, an engineer could quickly identify critical load bearing elements. Therefore, if the computer is going to perform evaluations like these experts, it is important to establish what exactly the computer "sees" when it looks at digital graphic data.

V.B. Unscaled Graphic Information

Photography is by far the most common means of recording heritage structures. Photographs can capture a large amount of building information quickly and inexpensively. They can capture certain types of information that is not as readily, or even possible, to communicate with written descriptions or drawings. Photography is particularly useful in revealing three dimensional qualities, spatial relationships, condition, texture, scale, and context than line drawings. Certain technical or aesthetic qualities are also more efficiently captured in this way, such as complex ornamental features or structural details.

Once images are recorded onto film or video tape the next step in creating a digital photographic record is to translate them into a machine-usable digital format. This is achieved through either one of two processes: scanning (photographs and documents) or frame grabbing (video). Both processes require special equipment and software. The use of a Charge Coupled Device (CCD) still video cameras eliminates this step because images are recorded (and verified) on site not on film, but digitally on floppy disk based on input from a sensor (Figure V.1).

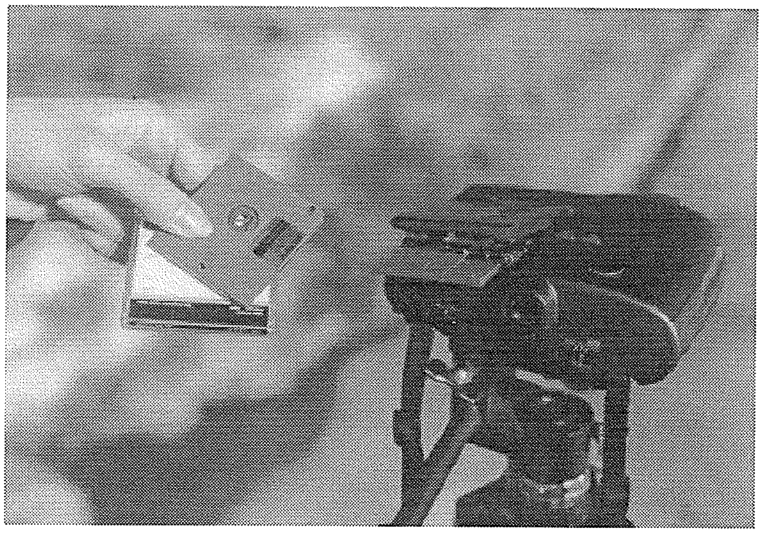


FIGURE V.1. - Canon CCD Camera

Scanners sample light intensity across a surface (i.e. a photograph or slide) and convert this information into an array of integers in computer memory, where each integer specifies a particular intensity. Frame grabbing is a similar process except it samples from an interval of a video signal. The more precise the sampling, usually described in terms of *dots per inch,* the more accurate the representation of the original. The location of each integer is organized on a square *raster grid* made up of smaller elements known as *pixels* or picture elements. This information is then translated into a visible display of corresponding intensities on an output device (Mitchell and McCullough 1991, 73).

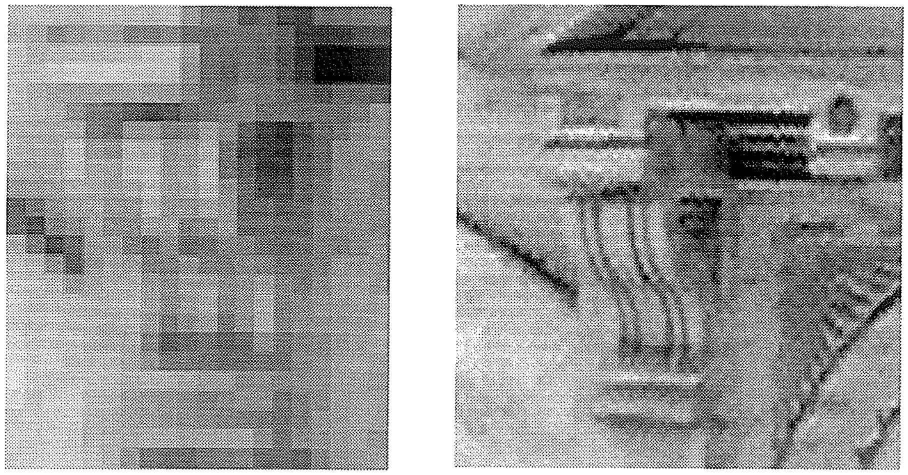


FIGURE V.2. - Raster grid & bitmapped image

Such digital images are known as *raster* or *bitmapped* (Figure V.2.). One of the primary characteristics of raster images is that they use a lot of computer memory because they store information about each pixel that is in the image. Each picture element represents at least one bit of memory per pixel (on=1, off=0). Zero represents intensities approaching or equivalent to white and one represents black. By assigning more memory to each pixel it is possible to create higher resolution images that include intermediate tones. A completely smooth-looking image is achieved when at least eight bits per pixel are used, which gives a scale of 256 tone gradations (Mitchell and McCullough 1991, 76). The more memory allotted to the image files, the more powerful the computer needs to be to work with as well as store them. Raster images also have to be stored on media capable of handling large amounts of data such as hard drives, optical drives, or magnetic tape.

Achieving good image resolution is, however, important because significant visual information can be lost. Maintaining information in these files is essential if the digital image is the sole form of documentation for certain elements that, for practical reasons, would not be measured or drawn (such as complex ornamental details). Low resolution images tend to flatten out and loose their three-dimensional quality, having a chunky appearance. Therefore, the subtle quality of architectural form, materials and details are lost (Figure V.3.). In the case of heritage buildings where their significance is embodied in such visual qualities this information is critical. Higher resolution images also provide the opportunity for different types of raster analysis and processing.

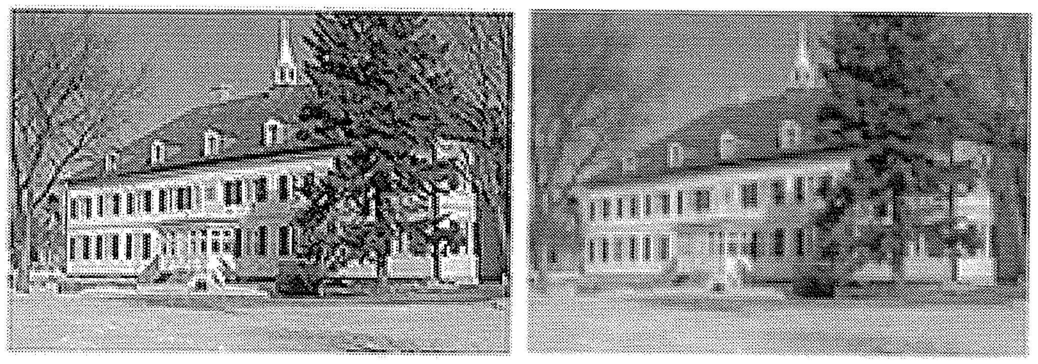


FIGURE V.3. - Comparison of image resolution

During the redevelopment process, a photographic record is produced in the initial stages of data acquisition and planning. With this first record a designer is able to visualize the site, buildings and key features off site. In the final design stages photographs are used to complement measured drawings and histories as part of the heritage record.

Digital photographic records are important to include as part of a digital heritage building file because of the role they play in visualization of different issues for the designer. The greatest strengths of raster image records are for the purposes of reference and visual verisimilitude. However, all of the building information contained within a raster image is interpreted by the viewer of the image. Despite the many interesting effects and compositions possible through image processing, raster images are made up of unrelated sub-elements and are intrinsically two-dimensional and this ultimately limits their utility for interpretation by both the designer and computer. In a raster image the designer cannot adjust their vantage point or move around the objects presented, nor ask the computer to do things like show the effect of shadows at different times of the year. Attaching attributes, such as the name "window," to individual pixels is simply not practical. In order to perform the complex analysis required in the redevelopment process a more complete digital model that includes geometric and semantic information about the heritage building is required (Figure V.4.).

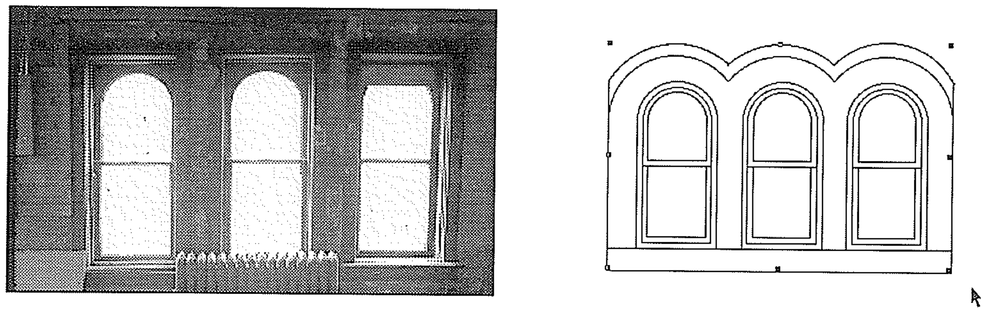


FIGURE V.4. - Raster and CAD drawing of the same window object

V.C. Scaled Graphic Information

Measured drawings for historical buildings include: location plans, plot plans, floor plans, elevations and so on. In the context of redevelopment, measured drawings eventually evolve into contract or working drawings. Of the drawing types cited, the floor plan is traditionally the most commonly used document for analysis.

Computer Aided Design programs provide tools for drafting measured building information on the computer. All drafting software is based on certain assumptions about architectural drawings. One assumption is that architectural or engineering drawings can be constructed from three basic primitives: the point, line, and arc. Drafting software is based on a two dimensional rectangular Cartesian system. Pairs of numbers are used to specify end point coordinates of lines on a plane defined by x and y axes. These numbers are stored in binary format in memory. Drafting systems employ eight, sixteen, or thirty-two bit binary numbers to represent coordinates. Coordinates are represented to a finite precision because of the limitation of memory, but even eight bit drawings are usually accurate enough for most applications.

Instead of entering integer coordinates, most CAD environments allow the designer to work in units of measure, like meters and centimeters, that are converted internally. In CAD files all of the information is represented mathematically. As a result building data can be stored at "full size" and changing between units of measure and display/printing scales is readily achieved. In terms of drawing heritage buildings, units of measure can be adjusted to match those used at the time and the place that a building was built. Building materials and organizing systems relate back to the standard unit of measure of the locality whether it is the English foot or the Japanese Ken. Using the original units make it easier to draw the structure. Using a CAD file allows the designer to easily switch back to a term of scalar reference that is more familiar.

There are a variety of techniques for entering coordinate points into the computer including: keyboard, digitizing tablet and light pens, but the most common is with a mouse. Using a mouse allows the designer to input the first point of a line by positioning a cursor on the screen, clicking and then *rubber-banding* to enter another point thereby establishing origin, direction and length. More complex shapes are created from a series of smaller line segments (polyline) to form open or closed polygons. A curve is also represented in this manner, where the visual smoothness of the curve is determined by the number of segments used. This type of digital drawing, because it is made up of lines, is often referred to as a *vector graphic* (Figure V.5.).

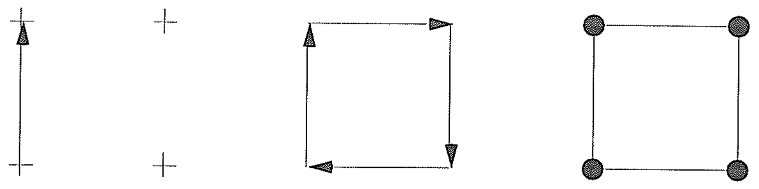


FIGURE V.5. - Vector graphics - square

Vector graphic files are considerably smaller than raster graphic files. For a line in a raster drawing each pixel location and value along its entire length is independently stored and manipulated. In a vector graphic file only coordinates of the end points of lines are put into memory with the connecting elements calculated for display and output. Correspondingly, it is also easier to modify and move an individual line in a vector graphic because the changes to the end points effect the whole line.

From the architect's point of view a lot of the advantages to CAD vector drawings are realized through the fact they are easy to modify and information can be reused within the same drawing or to create others; such as duplication of a single window to create window patterns or using the floor plan information to create a reflected ceiling plan. These drawings communicate to the designer as much information as any two dimensional traditional paper drawing does. However, on the computer, vector drawings contain purely geometrical data. As a result what is explicit to the computer is a very limited concept of architecture as collections of lines and boundaries, without a more comprehensive understanding of what exactly it is that these lines are defining.

V.D. Object-based CAD Graphics

Object-based CAD systems treat graphic entities as objects. Such systems have imbedded knowledge about the geometric properties of a given object, and allow the object to be linked to a variety of attributes that are not necessarily geometric properties. In object-based CAD, object classes may be defined by purely geometric attributes (e.g. squares), but can capture inherent attributes such as "doors" or "windows" or combinations of symbolic attributes such as all "oak doors with leaded glass."

Computationally, an object-based CAD system uses a sophisticated representation method that contains a deeper description about the characteristics of graphic entities. For example, if one draws a graphic entity such as a rectangle, the raster environment will treat it as an arrangement of unrelated pixels, the vector environment will treat it as a set of lines and coordinate locations (although other attributes may be tagged to vector entities), and the object-based environment will recognize a *rectangle object.* A rectangle object is understood to be a four-sided entity with parallel sides, at right angles to one another (Hill, et al. 1993) (Figure V.6.).

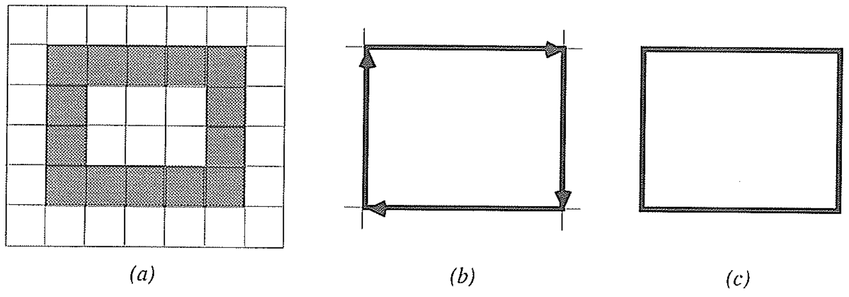


FIGURE V.6. - (a) raster, (b) vector, (c) object graphic

The unique geometric characteristics of the rectangle object are represented by *parameters*. "*Parametric variation* is possible where part of the data structure used to describe a given object in a CAD drawing contains variables instead of constants and the values of some variables are dependent on the values of others," (Mitchell and McCullough 1991, 117). In the example of the rectangle object the value of parameters for the "height" and "width" for the object instance are supplied by the user.

Object-based CAD environments will also support object to object parameters to varying degrees. These parameters define how change to the geometric variables of one object effects the other. Relationships between objects can be as simple as sharing a common coordinate point to being quite complex. In the case of classical proportioning, for example, the diameter of a column determines the dimensions of the shaft, capital, pedestal, and the entablature. Maintaining this type of complex relationship is very difficult because the computer must be able to not only recognize relevant objects in a drawing (i.e. column, shaft, pediment), but also understand the parametric constraints of classical proportioning. This feature, although it is not currently found in commercially available CAD packages, relies on the ability to relate knowledge with distinct graphic entities that is characteristic of object-based graphic environment.

The spatial relationships between objects are a significant part of architectural design that can be determined from geometric information. Object-based CAD descriptions, because they contain the dimensions and locations of objects (such as rooms), can be used to infer the topology of the design (Coyne, et al. 1989, 96). *Topological analysis* shows the spatial relationship between elements that is useful for describing such things as the organizational system of a historical building layout or generating bubble diagrams for new use patterns (Figure V.7.).

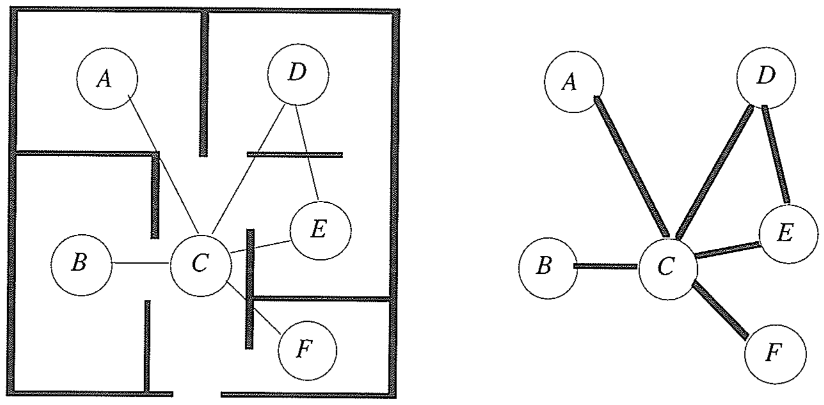


FIGURE V.7. - Semantic network representing spatial relationships (floor plan and topological diagram)

Although this study is concerned primarily with two-dimensional CAD building representations (plan, elevation), three-dimensional CAD is the best way to model vertical and horizontal relationships (volume, vertical circulation).

V.E. Creating Object-Based CAD Heritage Building Records

The drawing tools in the object-based CAD component of the Smart Graphics Environment are used to generate and modify the drawn elements of the heritage building design. As previously mentioned, because they take a long time to produce, measured drawings the most expensive form of heritage building documentation. However, there are a number of opportunities where instead of inputting all of the information directly the computer can be used to help facilitate the drawing process. In most cases this involves tools used to extract coordinate information imported from other raster and vector-based CAD programs and be overlaid with object information. The following subsections discuss data acquisition approaches as well as setting standards for: drawing accuracy, graphics, and layering in object-based digital heritage building records.

V.E.1. Data Acquisition. Typically the source of the measurements for the drawings is collected from physical evidence recorded on site. Dimensions for measured drawings normally come from three types of records: documents, hand measures and photographs. Documentary sources represent measurements that someone else has recorded including: original or alteration drawings, historical photographs, published accounts, previous surveys, specifications, and so on (Burns 1989, 125). Of these types, hand measuring the building, if executed properly, produces accurate results. This technique, however, requires careful planning and coordination. It is labor intensive and, therefore, expensive. The larger and more inaccessible the building, the more difficult it is to obtain a complete set of measurements efficiently and economically by hand.

Where existing plan or elevation drawings are available it is possible to make a scan of the drawings and trace them using object-based CAD tools. It is important to remember however, that information that can be gathered from original drawings may not be accurate and detailed enough for the purposes of redevelopment. Original drawings can be merely sketches or do not represent *as-built* conditions due to changes made during or after construction.

Acquiring drawing information from photographs can be a feasible alternative to hand measuring or scanning original drawings. Architectural photogrammetry, which combines the principles of photography and geometry in order to extract measurements from photographs, is used in a number of different ways with the computer to create CAD drawings. Rectified photography, where perspective distortions are eliminated in the photographic raster image, can be achieved using software programs such as Digital Video Rectification [DVR]. This allows measurements to be taken directly from the photographic image. Specialized software programs like CAD Overlay™ combine raster image processing tools along with CAD tools. This allows the user to adjust the contrast and brightness of a bitmapped image so that it is easier to trace.

Computer Aided Reverse Perspective Analysis [CARPA] developed by (Crankshaw 1990) can be used to generate building plan footprints from photographic views of rectilinear objects. The computer achieves this by reversing the geometric principles used to create perspectives by projecting lines from plan views, and applying them to perspective photographic views of the building.

The examination of the relationship of equivalent and adjacent integers has been used to develop *Optical Character Readers* [OCR] capable of automatically recognizing characters on pages of printed text, making them usable by word processors. This eliminates the need to retype existing text into the computer (Mitchell and McCullough 1991, 96). Recent research in automatic architectural shape recognition (Tan 1990) (Nagakura 1990) (Koutamanis 1992) is based on the concept of matching of patterns of pixels or *templates* to instances within a bitmapped image (similar to how an OCR looks for patterns of that make up a text character). Similarly, edges of objects are recognized by the occurrence of adjacent equivalent pixel values.

Generic object matching and edge recognition in raster images are particularly significant to semi-automatic measuring techniques where feature extraction algorithms are assisted by input by the user. The Digital Photogrammetry and Architectural Design [DIPAD] system described in (Strelein, et al. 1992) combines edge recognition with photogrammetric techniques and is capable of extracting accurate three-dimensional (x,y,z) locations for points, lines, open polygons, closed polygons, and surfaces from multiple photographic views. These in turn can be exported in *Data eXchange Format* [DXF] into an object-based CAD environment for further processing.

To date, a practical system that is the equivalent to an OCR (totally automatic translation graphic images into a CAD format) has not yet been developed. This is because it is difficult for the computer to recognize even standard architectural entities, such as door swings or windows, because, unlike typewritten characters, such architectural elements are integrated with other graphic entities (i.e. wall) and are not distinct from their neighbors. Totally automatic recognition of architectural objects within photographic images is an even more difficult proposition, where it is hard to anticipate certain patterns of pixels, often in perspective, as being related to particular object or class of objects. A particular form of Artificial Intelligence system, neural networks, has been shown to be quite effective in identifying patterns that it "learns" - to the extent of successfully distinguishing between male and female human faces. In the future this technology may also prove to be a means by which bitmapped images can be read. Other techniques involving robust statistical analysis of geometric data to extract geometric primitives as described in (Roth and Levine 1993) may also prove effective.

V.E.2. Drawing Accuracy. In order to assist in entering information most drafting systems allow the user to define modular orthogonal grids parallel to the coordinate axes. The cursor can then be set to *snap* to the user-defined grid. In a CAD environment walls can meet at precisely ninety degrees and can be continuous along their entire dimension. These characteristics are obviously desirable in designs for new construction. However, in the case of drawing heritage buildings issues of drawing precision and accuracy are at odds with this type of environment. In reality buildings are not built with this kind of precision and over time, with use and exposure to environmental factors, further deformation and shifting occurs.

Measured drawings of historical buildings for redevelopment purposes do not differ significantly from typical architectural contract or working drawings except for the fact that they represent *as-built* condition. A preliminary plan can be accurate up to within ten cm (4 inches) whereas a final detailed record is expected to be within twenty-five millimeters for building plans and five millimeters for details (Letellier, et al. 1993, 8). High accuracy is important because subtle changes are revealed that would otherwise remain hidden, such as deformation of walls, that could be indicators of structural damage or deterioration.

It is because of the irregularities of heritage buildings that when entering the coordinates of the perimeter walls of rectangular room CAD operators often find that the last and the first points do not match, resulting in a *closing error* (Figure V.8.). Closing errors should always be annotated on CAD drawings. The occurrence of closing errors can be reduced by taking diagonal measures on site. A *total station*, or digital theodolite, records surveyed points directly into a digital form. Imported into the CAD environment, these can provide control point coordinates for snapping to and for verification of hand measures.

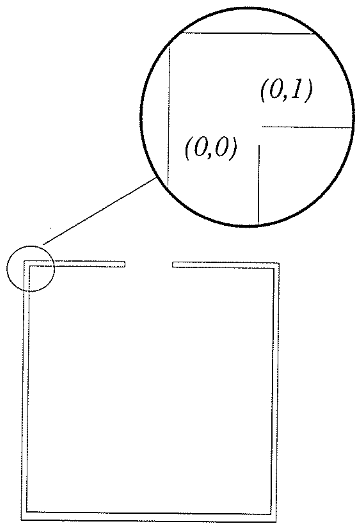


FIGURE V.8. - Closing error

V.E.3. Graphic standards and layering. Graphic standards for the production of traditionally drafted architectural and engineering drawings have existed for a long time to help ensure clear communication of ideas. Heavier line weights are used to differentiate between objects in front of others, or a line and arc are commonly used to show direction of the swing of a door. Usually such standards are customized to suit the design offices in which they are used. The desire to achieve a universal exchangeability of digital heritage files, particularly between domains, requires developing consistent approaches for the use of: line weights, lettering, dimensioning, hatching/fills, drawing sheet sizes, drawing scales, symbols and formats. Presently, it is difficult to identify a single industry CAD graphic standard. Two recent proposals include (McGonigal, 1993) and (CSA, 1993). As a general rule CAD graphic standards for heritage files should be as simple as possible, be set up before drawing begins, and be well-documented.

One of the very first standards to set for a CAD drawing file is the use of *layers,* sometimes known as levels. This becomes the very first level of classifying CAD object information.

The concept behind layering is based on the use of transparencies or tracing paper to overlay design information. In the CAD environment each layer of information can be edited, displayed and/or printed out individually or in combination with others. In this manner certain aspects of a design can be studied in isolation without extraneous information obscuring or overcomplicating the drawing. Furthermore, the use of layers reduces the redundancy of information. A wall drawing layer only has to be drawn once and then can be used in combination with other layers to create mechanical, electrical, plumbing, and architectural plans.

Layers should be set up before any drawing takes place. Arranging and naming layers in a CAD file should be according to an accepted industry standard. CAD layering standards need to take into consideration the division of data into:

(1) construction assembly separation (walls, doors, windows);

(2) drawing separation (organizing layers for output);

(3) location separation (where in the building i.e. ground floor, etc.);

(4) project scope separation (phasing, alternate designs, historical vs. modern renovation); and

(5) three dimensional separation (*Electric Architect* 1991).

There are several CAD layering guidelines currently available, although no single standard has yet to be universally adopted. Two North American guidelines, (AIA 1990) and (Public Works Canada 1990) share essentially the same approach to layering. Both divide data into major and minor groups, offer long and short versions, are open ended (allow for user-defined fields), and can be easily used across a number of different applications. Using the guidelines as a starting point it is possible to propose a layering strategy to address the specific needs of restoration and redevelopment in heritage buildings.

Each layer name contains alphanumeric characters identifying a major group or category, minor group and a modifier. Major groups of data include:

(1) A - Architecture, Interiors and Facilities Management

(2) S - Structural

(3) M - Mechanical

(4) P - Plumbing

(5) F - Fire protection

(6) E - Electrical

(7) C - Civil engineering

(8) L - Landscape architecture.†

Minor groups are used to subdivide major groups into more specific classes of information. These groups tend to relate more to the separation of plan information than non-plan information. Several layers are used to describe a single floor in a building whereas only one or two are used for an elevation. This is because plan information is more often shared between disciplines than non-plan information. Therefore, the separation of plan data is finer. In the case of heritage building records used in the Smart Graphics Environment the elevation may be further divided to suit the needs of object identification (Figure V.9.).

† Public Works Canada divides major groups into those for the building sector and heavy civil engineering. In addition to the eight groups identified, PWC adds for buildings: I - Interiors, R- Reality and space management, U - Security, W - Communications.

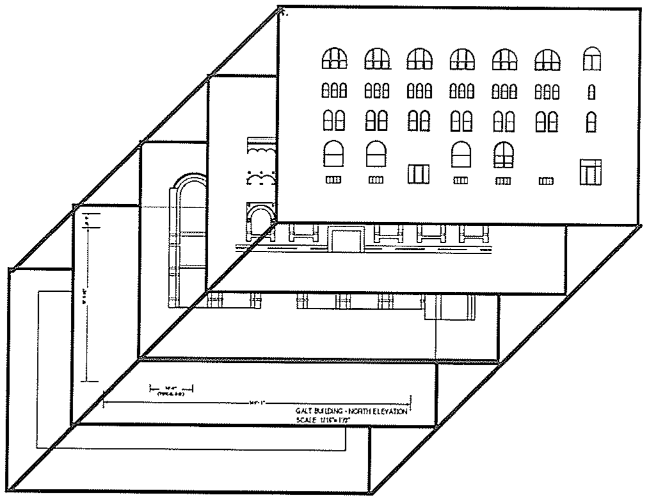


FIGURE V.9. - Layering of elevation object information

Minor groups contain either building information or drawing information. Building information layers are often shared between drawing groups. They contain data relating to the physical form of the site, building, or objects within the building. Examples include walls, doors and windows. Drawing information layers are generally connected to a single layer only and contain annotations, dimensions, and cross references (AIA 1990, 15). Drawing information layers that do appear in almost all heritage building drawings include: (1) reference grid (established by control points surveyed on site) and (2) border and title block linework. An example of a layering strategy for a simple architectural digital heritage record based on AIA guidelines is shown in Table 2.

TABLE 2 - Architectural CAD drawing layers for a heritage file

Layer Name	Short Form	Layer Contents
A-WALL	AWA	walls
A-DOOR	ADO	doors (jambs, casework, swing)
A-GLAZ	AGL	windows, window walls, glazed partitions
A-FLOR	AFL	floor information
A-CLNG	ACL	ceiling information
A-ROOF	ARO	roof
A-ELEV	AEL	interior and exterior elevations
A-SECT	ASE	sections
A-DETL	ADE	details
A-SHBD	ASH	sheet border and title block line work
A-GRID	AGR	grid established by surveyed control points
A-PFLR	APF	floor plan
A-PCLG	APC	reflected ceiling plan
A-PHOTO	APH	photographic key plan
A-PXFU	APX	fixtures and furniture plan
A-PDEM	APD	demolition plan

CAD layering conventions achieve their open-endedness by attaching modifiers (two to four characters) to the layer name that are either standard or user-defined. These can be used to indicate floor levels (i.e. -01, -02 . . .etc.), or even the language of the text used (i.e. "-E" for English). The suffix modifiers "-EXST," "-DEMO," "-NEWW," "-PLAN" differentiate between layers of existing items to remain, existing items to be demolished, new work and planning or design information respectively. The layer name is then put together:

A-WALL-01	for walls on the first floor.
A-WALL-DEMO-01	for walls to be removed from the first floor

Although it is important that the layer name contain a clear indication of its contents many users prefer, or may be limited by their CAD system, to use a shorter form:

A-WALL-01	AWA-01
A-WALL-DEMO-01	AWADE-01

Special modifiers are required to indicate unique layers specifically relating to heritage building records:

-PHOT	PH	photographic key plan
-IMAG	IM	photographs (raster images)
-CFIG	CF	special groupings or configuration of entities
-CLSG	CL	closing error notations

A photographic layer is used for photographic key plan (exterior or interior) to indicate camera positions. The actual raster images associated with the drawing file can be imported temporarily to a designated layer for viewing or tracing. Storing raster images permanently within the CAD file would make it overly large. Configuration layers are used to organize object to object spatial relationships. Elements of a historical building have individual characteristics as well as those attributable to being a part of a particular arrangement or grouping. The historic value of a window pattern, for example, is sometimes greater than the value of each window in isolation.

As mentioned in the beginning of this section, layering is the first level of classification of graphic entities in a CAD file. They are designed to assist in the division of data for management of project files. Care has to be taken to ensure that the layering strategy employed does not become overly complex. Too many divisions make a drawing file very difficult to create and maintain. Using unique colour or line types for layers does not translate well between systems. Inasmuch as layers could be used to communicate very specific information about a given entity, this approach is simply not practical. A layer containing doors is far more relevant to most applications than numerous separate layers consisting of: hallway doors, bathroom doors, entrance doors and so on. Using attributes is a much more flexible approach to assigning non-graphic (such as door type) information.

V.F. Object-based Database and Architectural Attribute Coding

In the Smart Graphics Environment the object-based CAD component serves as the interface between the designer and the KBS. This allows the designer to receive critical feedback directly within the CAD drawing environment. The particular advantage to using object-based CAD heritage building files is that it allows the designer to link specialized kinds of information needed by the KBS to provide this feedback with recognizable objects in a graphic.

An *Integrated Graphic Database* [IGDB] associates information with objects in a drawing. In general, a database contains one or more data file(s) (text, graphic, and/or numeric) "however defined, accessed, or stored that holds non transient data in a computer application" (Oxford Dictionary of Computing 1991, 110). An integrated database combines information previously held in many separate files. An Integrated **Graphic** Database refers to a particular type of database that draws its information from CAD drawing files as well as information provided by the user. Based on these definitions of databases it follows that an object-based IGDB holds data files relating to objects in object-based CAD drawings.

A CAD system that incorporates a database feature, as many now do, will have a *Database Management System* [DBMS] that offers control over the data, file organization and access methods. A sophisticated database manager integrated with a CAD system dynamically links information in the drawing with information stored in the database. This means that any changes made to a graphic entity in the drawing (such as increasing the height) are immediately updated in the data file associated with that particular object.

The purpose of the IGDB component of the Smart Graphics Environment is to handle the manipulation, exchange, and linkage of information between the user, CAD drawing objects, the object *Architectural Attribute Code* records [AAC], graphic files, and the KBS objects. The objects in the CAD drawing are stored as *object instances* in the IGDB.

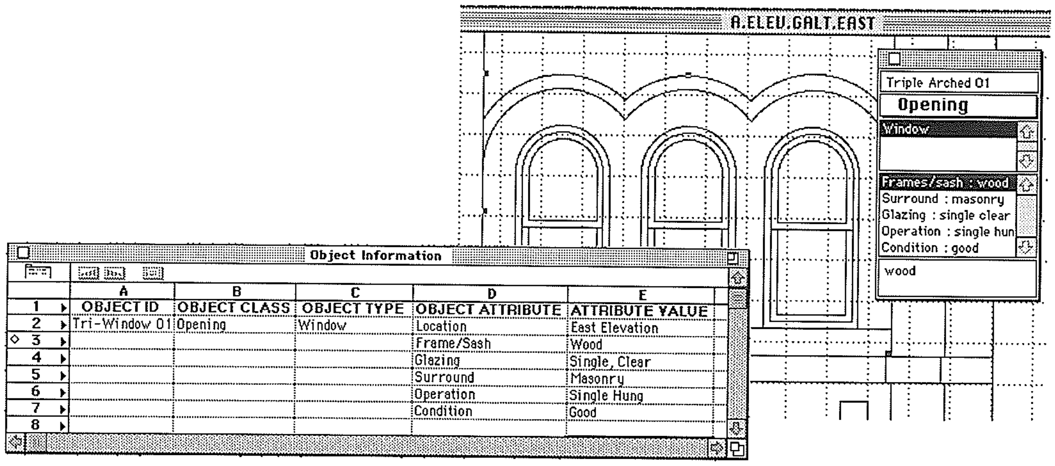
What is identified as an object and how it is defined is very much dependent on the analysis the computerized "expert" is being asked to perform. A structural engineer interested in performance will analyze a column based on information about the material, height, width and loads. A historian concerned with identifying character-defining features on an elevation examines data relating to age, context, appearance, and notes unique colour, shape or decoration. Facilities managers investigating new occupancies are interested in the form and organization of spaces and activities. It is for this reason that the techniques or language used to describe design is flexible enough to accommodate a variety of object descriptions that arise from different architectural design domains.

Component objects of a historical building design can refer to physical elements (door, window, wall), spaces (room, corridor), or concepts (fire separation, node). The purpose of *Architectural Attribute Coding* [AAC] is to describe the properties or characteristics of architectural objects. Object attributes can be:

• physical;	what it is (dimensions, colour, mass)
• functional;	what it does (opens, flushes, divides)
• signal;	what it represents (Georgian, symbol of life, age)
• topological;	relations with other objects (is a type of, is adjacent to)
• geometrical;	how it is shape/orientation/measure (parallel sides, triangular, cube)

Architectural concepts tend to be hierarchical in nature. This allows for an economy of attribute coding because it is possible for objects to inherit properties shared by generic *parent* classes of objects. For example, all doors belong to the class "Door" and therefore share common attributes of: material, width, height, thickness, and finish. A subclass of "doors" may differentiate between doors with windows in them, or by their operation such as bi-fold, sliding or revolving. Each individual object instance will have attributes that are not shared by its parent class and therefore is also identified by a unique *name*. For instance, a uniquely paneled period door located at a principle entrance may be identified as "Door-1" of the class "Door."

The AAC is stored in the IGDB in a *frame* which is made up of attribute records and fields. Attribute records contain groups of fields. Fields contain attributes and *attribute values* attached to each object. For example, a window object with the attribute "frame material" can have an attribute value "wood" (Figure V.10.). Predefined records and fields contain information associated with object classes. For instance, the attribute record for the "Window" class will include information on material, glazing type, finish, etc. Additional attribute fields can be added to further clarify the object description. CAD database systems offer different techniques for customizing records and entering/manipulating field information. This includes special data palettes and/or separate database or spreadsheet environments that are used within the drawing environment.



Triple Arched 01
Opening
Window
Frames/sash : wood
Surround : masonry
Glazing : single clear
Operation : single hun
Condition : good
wood

	A	B	C	D	E
1	OBJECT ID	OBJECT CLASS	OBJECT TYPE	OBJECT ATTRIBUTE	ATTRIBUTE VALUE
2	Tri-Window 01	Opening	Window	Location	East Elevation
◇ 3				Frame/Sash	Wood
4				Glazing	Single, Clear
5				Surround	Masonry
6				Operation	Single Hung
7				Condition	Good
8					

FIGURE V.10. - Integrated Database, data entry palette, and CAD object

The sources of attribute values can be categorized as either:

(1) explicit,

(2) calculated,

(3) assigned, or

(4) default.

Both explicit and calculated attribute values can be extracted directly from information contained in a CAD graphic. As described previously, object-based CAD graphics use both fixed and variable geometric parameters to control how the objects are displayed and manipulated. Therefore, a designer automatically assigns particular explicit geometric attribute values by creating an object in a CAD environment. For example, an object drawn as a square automatically will have sides of equal length.

Calculated attribute values are *variable dependent*. They are derived from the values of other object attributes using procedures imbedded in the IGDB. For example, the amount of paint required to finish a rectangular door can be determined from a procedure involving the calculation of the door's "area" attribute value based on the value of two other attributes, "height" and "width".

Assigned attribute values are supplied by the user to objects to describe characteristics that are not described graphically, such as historical value or finish. Default attribute values are automatically assigned by the system to cover information that is missing or incomplete so that some level of analysis can still occur. These values can be inferred from the rules required for analysis of the building or assumptions based on heuristic domain knowledge. For example, the default clear width for all doors may be based on an average width of 800 mm unless otherwise specified. Assigned and default attribute values are *variable independent* in that they are not derived from the values of dimensional or geometric information contained in the CAD drawing. Attribute values for "class" of object can be derived from the CAD layer designation of objects in the drawing (i.e. objects on the "A-DOOR" belong to the class "door").

V.G. References

American Institute of Architects [AIA]. 1990. *CAD Layer Guidelines.* M.K. Schley (ed.). Washington.

Burns, J A (ed.) 1989. *Recording Historic Structures.* National Park Service. Washington, D.C.: American Institute of Architects Press.

CAD Overlay ESP™. (computer program) Image Systems.

Coyne, R., M. Rosenman, A. Radford, M. Balachandran, J. Gero. 1990. *Knowledge-Based Design Systems.* Sydney: Addison-Wesley Publishing Company.

Crankshaw, N.M. 1990. "CARPA: Computer Aided Reverse Perspective Analysis." *APT Bulletin.* XXII No.1/2. 117-131.

Canadian Standards Association [CSA]. 1993. *Computer-Aided Design Drafting (Buildings), B78.5-93.* Ottawa.

Hill, S., M. Evans, L. Strachan, P.R. Perron. 1993. "The use of smart graphics for the redevelopment of historical structures." presented at *Design & Decision Support Systems Conference.* 6 - 10 July 1992. Eindhoven, The Netherlands. [Pending publication in the Journal of Architectural Research and Planning].

Koutamanis, A. and V. Mitossi. 1992. "Architectural computer vision - automated recognition of architectural drawings." presented at *Design and Decision Support Systems Conference.* 6 - 10 July 1992. Eindhoven, The Netherlands.

Letellier, R., J. Bell, P. Sawyer, J.P. Jerome. 1993. "New low-cost computer tools for heritage conservation practices & built environment sustainable development programs." (Training Course Handout) 15 - 18 November Ottawa: Heritage Recording and Technical Data Services, Architectural and Engineering Services, Government Services Canada.

McGonigal, G. 1993. "ARIDO defines CADD graphics standards." *Canadian Facility Management.* October. Don Mills: CFM Communications. 24 - 25.

McKee, H. 1970. *Recording Historic Buildings - the Historic American Buildings Survey.* Washington: U.S. Government Printing Office.

Mitchell, W.J.and M. McCullough. 1991. *Digital Design Media.* New York: Van Nostrand Reinhold.

Nagakura,T. 1990. "Shape recognition & transformation: a script-based approach." *Electronic Design Studio.* M. McCullough, W.J. Mitchell, and P. Purcell (eds.). Cambridge: MIT Press. 149-170.

Oxford Dictionary of Computing. 1991. New York: Oxford University Press.

Public Works Canada. 1990. *CADD Layering for Architectural and Engineering Applications.* October. Ottawa: Architectural and Engineering Services, Public Works Canada.

Roth, G. and M. Levine. 1993. "Extracting geometric primitives." *CVGIP: Image Understanding.* 58 (1), July. 1- 22

Streilein, A., H. Beyer, T. Kersten. "Digital photogrammetric techniques for architectural design." presented at *XVII ISPRS Congress*, August 2 - 14. Washington. XXIX, Part B5 *International Archives of Photogrammetry and Remote Sensing* . 825-831.

Tan, M. 1990. "Saying what it is by what it is like - describing shapes using line relationships." *Electronic Design Studio.* W.J. Mitchell and M. McCullough (eds.) Cambridge: MIT Press. 201-214

"______". 1991. "CADD layering standards: a complex issue." *Electric Architect.* 2 June/July. Ottawa: Royal Architectural Institute of Canada. 2-4.

CHAPTER VI

THE DESIGNER

This chapter is used to explain the role of the Designer in the Smart Graphic Environment. The advantages to using an interactive approach for design decision-making with KBS are revealed. Included in this chapter is an examination of the importance of user interface in facilitating decision-making. Expedient methods for communicating design data, initiating analysis, and presenting feedback to the user are discussed in the context of a *client-server* model. In this model, the designer communicates with other components of the Smart Graphic Environment without leaving the drawing environment.

VI.A. The Clever Designer

In his article, "A New Agenda for Computer-Aided Design," William Mitchell commented that CAD and database analysis procedures had not supplanted human critics. He pointed to the fact that buildings are complex entities which "perform subtle economic, social, and cultural roles" and that "these can only be understood adequately by reasoning about them in the light of extensive economic and social and cultural knowledge" (11). What makes computer intelligence *artificial* is that it is based on arithmetic and logic alone and because of this it lacks a broader contextual awareness. Anatol Holt, a researcher in artificial intelligence, noted that the computer's ability to perform brilliant chess moves while the house is burning down clearly does not show intelligence. Expert computer systems can be likened to an idiot savant -- exhibiting exceptional skill in a particular field of expertise but lacking common sense. What makes expert systems truly useful is that they make "clever people cleverer" (Browning 1992, 6).

Creating a Smart Graphic Environment is only in part about developing digital design data and expert knowledge-base systems. The end-goal truly is to give the designer tools to do a job they already do, only better. No matter how powerful the computer tool is, taking advantage of it still requires the designer to be skillful. Learning how to collaborate and work with other people to make decisions can be hard enough, learning to decision-make with a computer is harder still (Browning 1992, 24). As a developer of an intelligent design environment one has to be mindful of the strengths and weaknesses of both the designer and computer systems and find ways to use the skills of one to augment the skills of the other.

The Smart Graphic Environment was intentionally conceived as an interactive one. The role of the designer is to do the tasks that computer systems are not very good at doing. By taking advantage of the talents of a human designer, this approach is much more efficient in the solution of re-development design problems. On the one hand, computer systems can be relied on to handle, sometimes simultaneously, all of the facts and highly specialized expertise relating to heritage buildings and redevelopment that is built into them without forgetting or becoming overwhelmed. On the other hand, it is the knowledgeable designer that provides the experience and skill required to use design knowledge and data wisely and creatively to resolve design problems.

Within the Smart Graphics Environment the role of the designer is to:

(1) enter design data;
(specify or limit the context of analysis),

(2) initiate evaluations;
(structure problem within domain of discourse),

(3) interpret feedback provided by KBS;
(reason with intuition and inference based on broader context)

(4) resolve design problems by eliminating errors;

(5) recognize desired design state (design completion).

VI.A.1. Entering Design Data. In order to come to consensus, decision-makers have to share a common understanding of the problem and be given all the information they need to solve it. As we have discussed earlier in this study it is easy to collect and generate lots of digital heritage building records in all kinds of forms and formats. Moving from a merely digital description of visual and textual information, in terms of pixels (graphics) and alphanumeric characters (text), to recognition of building elements and understanding human language (*natural language*) simply cannot currently be done efficiently by a computer. A strength of an expert system is that it is able to deal with large amounts of information. Interpreting data in a meaningful way to solve problems, however, is quite another matter.

Traditionally, a designer performs a juggling act involving combinations of different abstract building representations (photographs, drawings) and the interpretation of voluminous functional and performance specifications, including such things as building codes and guidelines for maintaining historic integrity and the client's own functional program. All of this must be done so that designers can determine whether their designs establish an appropriate fit between codes and historic preservation, while still meeting the necessary design criteria for new use.

The Smart Graphics Environment is intended to allow for the development of a building redevelopment design using conventional CAD tools. However, to truly take advantage of the system certain design protocols need to be introduced. Elements required by the expert system must be identified in the CAD application and certain attributes have to be assigned or system defaults will be used. Information is also organized in accordance to CAD drawing standards, including the use of layering. Photographic information (raster files) tell a story about building materials and condition. Such information must also be attributed to the instance of those geometric entities represented within the object-based CAD environment.

The identification of objects must reflect the needs of the knowledge-base system's testing mechanisms, whether they are about establishing code compliance, historical classification, or functional evaluation. Accordingly, conditions are placed upon the designer to define elements in certain ways depending on the context of the evaluation. For example, if the area of a room is required for testing, then the object representing the room should be of an appropriate type, i.e. polygons, rectangles etc., rather than drawn as a series of unconnected walls. In this way the user is considered to be an integral part of the Smart Graphics Environment as a kind of interpreter. It is much easier to teach a human designer to describe design in a way the computer can understand than vice versa. As a result, to some extent, the environment determines how the designer proceeds and how information is generated. Such constraints may prove to be too limiting for some designers, however, for others the benefits of interactive expert evaluation will outweigh the constraints on design representation.

In time, as CAD drawing environnments improve, the tools used to create architectural object representations will not only be more functional and flexible for all designers, but also have a understanding of the nature of the object that is being drawn. Commercially available CAD systems already have begun to develop such features as a "wall tool," symbols with imbedded attributes, and context sensitive cursors (for aligning objects).

VI.A.2. Problem Structuring. The design process is characterized by goals that are not generally well-formed at the outset. By observing architects at work researchers have discovered that experienced architects show a great skill in *problem-structuring* or *puzzle-making* (Aikin, 1988)(Archea, 1986)(Coyne 1990). This means that architects have the ability to structure ill-defined problems and make them tractable. They do this by restructuring ill-defined problems into manageable well-defined parts that they resolve separately and then reassemble the partial solutions into a general solution for the entire problem (Aikin 1988, 178). As an architect works through sub-problems different design alternatives become clear. Design scenarios are addressed according to global constraints (entrance location, mechanical systems) first, since these have many dependencies, before local ones are resolved (hardware/fixture selections, furniture layouts).

In the Smart Graphic Environment the designer defines the sub problems for the expert system to solve. The knowledgeable user is one who knows what analysis needs to be done, and at what stage of the decision-making process. During initial design, as the designer starts to resolve global issues, they need not be reminded of conflicts with minor or detail constraints. Control over initiating evaluations means that the system is told by the user what to evaluate and when.

Expert systems are very literal-minded. They cannot do anything unless explicitly told how to do it. For example, unless it is taught to do so, an expert system will not recognize as unusual that a plan of a turn of the century house has a washroom, when most houses of this period did not have indoor plumbing. Carefully selecting knowledge to include in an expert system is important. Even though an expert system's knowledge is biased to a particular context or subject domain an expert system will try to apply its knowledge to everything it encounters. As a result of trying to anticipate what the KBS may come across, system developers run the risk of creating a profusion of rules to explain and clarify even the simplest of facts. Translating the entire history of architecture into a knowledge base, for example, is simply not practical. This means that expert systems work best when given a small part of a problem to work on that is simple and self-contained.

The expert system in the Smart Graphics Environment facilitates problem restructuring and resolution through critical analysis of design data provided by the designer. Expert analysis, initiated and controlled by the architect, is applied to different specific subproblems relating to redevelopment of heritage buildings that impact on design decision-making, such as establishing period classification and/or building code compliance. The knowledge base(s) therefore contain particular knowledge relating to solving subproblems, whereas the designer is responsible for resolution of the overall design.

VI.A.3. Interpreting Feedback. Whenever an expert system is asked to perform a design analysis it will attempt to apply its knowledge to the information presented to it and make conclusions. In the Smart Graphics Environment the feedback to the user can be in the form of identifying certain instances or problems, explaining the nature of such instances or problems, and presenting advice about how to proceed, such as suggesting how problems can be overcome or remedied.

Expert systems rely on logic to solve problems. This assumes that conclusions can always be reached in a consistent and predictable manner. This is obviously true of expressions such as "one plus one always equals two." However, in the real world one discovers there can be exceptions to any rule. This is particularly the case in design where, for example, many different instances can share the same functional classification. Consider how many physical variations there are that would qualify as instances of the concept "window" (Figure VI.1.). It is for this reason that conclusions reached by expert systems need to be couched in terms of *certainty factors* or statistical probabilities *(fuzzy logic)*. The designer themselves, aware of other issues or factors (i.e. cultural, economic, social, environmental), help to confirm the validity of any conclusion or criticism made by the expert system.

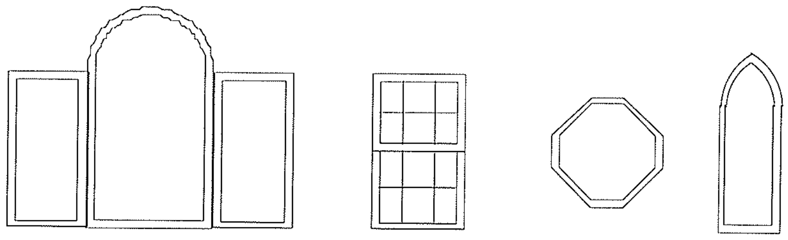


FIGURE VI.1. - Which graphic is the window?

In a design environment conclusions are often reached by analogy to other relevant precedents. One of the best problem-solving approaches to build into an intelligent design system is *case-based reasoning.* In this approach the expert system compares the current problem with others it has encountered and then uses the same problem-solving strategy. This is a particularly effective technique for problems that are straightforward and easy to identify. For example, conservation experts have a number of consistent strategies for dealing with problems common to all old buildings such as rising damp, or masonry repair. In this way expert systems can be used to give advice or present examples by matching current design information with precedents in its memory. The designer then has discretion to decide how best to adapt the information the system presents to them to their own design problem.

VI.A.4. Making decisions and resolving design problems. The Smart Graphics Environment is not intended to be able to automatically generate or modify existing heritage building designs to suit new uses. In the end, it is the designer who makes the decision as to how and what to modify in the design. Most generative rules based on different formalized languages of design are rigid and inflexible and are generally not equipped to deal with unusual or existing contexts.

Invariably every heritage building will have special or unusual features that make it distinct from all others, including those of the same period or style. In some cases heritage building features can be quite obscure; like the worn step at the entrance of the immigration office on Ellis island, or the unusual acoustic qualities of the whispering gallery of the dome of St. Paul's cathedral. Sometimes the sum of many parts contribute to a greater overall experience. Design synthesis is more readily achieved by an experienced designer who is better equipped with processes for responding, in a general way, not only to the feedback provided by the expert system but also the more subtle relationships a building has within its realm of particular community context, and human experience.

Inasmuch as the expert system is able to provide critical feedback, inevitably conflicts between different design goals of certain domains of expertise will occur. Here again the designer can bring additional criteria to bear in making a decision between opposing objectives. For instance, how would the system know when the preservation of a heritage window take precedence over security, cost or environmental control issues? In some historical districts maintaining the historical integrity of the exterior visual features is a critical issue whereas modification to the interior spaces may not be. Therefore the designer would have to consider more carefully design changes mandated by building codes or functional requirements that seriously impact on the exterior of the building than those that effect the interior spaces. To do this, the designer needs to have to control how different aspects of the design are evaluated by turning on or off certain object constraints in the knowledge base or by prioritizing design goals for the system to maintain. In this way the designer can concentrate on development of the design by correcting relevant errors.

VI.A.5. Recognizing desired design state. There are no absolute answers to design problems, but there are qualitative differences between "good," "better," and "innovative" design solutions. The function of the expert system in the Smart Graphic Environment is merely to assist the designer in narrowing down an appropriate solution space from a potentially infinite field of possibilities.

As the designer moves through the redevelopment design process within the Smart Graphic Environment, from the creation of digital heritage records, to expert analysis and then synthesis of new design solutions, their own understanding of the project increases. Better understanding of design goals and intentions for the project leads to an ability to make decisions with clear rules by which all actions are selected and judged as being appropriate or not. It is impossible to anticipate ahead of time, and imbed them into the rule-base of an expert system, a precise set of design requirements (*As-Required* design state) applicable to an entire building design. During a design process a designer will manipulate a design through numerous iterations, evaluating each time if more or less design goals are met. It is the role of the designer to recognize when all of the parameters of the desired solution space (established during the redevelopment process) have been satisfied by the proposed *As-Designed* state (after Mitchell, 1990b).

VI.B. User Interface

Enhancing human performance through the use of computer systems hinges largely on the ability to provide suitable *user interface*. As computers have become more and more a part of our daily lives research in the area of human-machine interface has become an entire field of study onto itself that delves into areas including human factors, cognition, and learning. The goal of such studies is simply concerned with making human users feel competent in using a computer system. Competency enables users to easily correct errors and accomplish their tasks (Shneiderman 1989, 167).

In developing a Smart Graphic Environment there are two issues to consider: (1) communicating with the computer system, and (2) presenting results to the user. Assuming the system has the proper functionality to do what the user needs, this functionality should be made easy to get at. This means facilitating such things as how the user issues commands, selects and initiates analysis by the knowledge base, and supplies information to the system. Presenting the results of the planning or analysis procedures in a timely and meaningful manner is obviously another important aspect of accomplishing tasks.

The approach taken in this study was to try to provide all of the interface between the user and the computer in one place -- the CAD environment. Whereas most architects may be experts in the field of building design, they are likely to be novices to computer concepts. Architects are used to communicating their ideas graphically through drawings. They are also familiar with receiving feedback through *red-lining* or annotations made directly on their plans or elevations. Therefore, the CAD environment is the most natural starting point for facilitating communication.

The designer's familiarity with drawing environments should help in: reducing the time required to learn the system, increasing the speed in which tasks are performed, reducing the number of errors made by users and improve overall subjective satisfaction. These are all measurable human factors goals that can be achieved through sensitive planning for user task needs with subsequent refinement through testing (Shneiderman, 1989).

The approach used for the Smart Graphic Environment is based on a *client-server* model in which the database and the knowledge-based systems act as server to the CAD system. In this approach all of the operations of the database and expert system are transparent. The following subsections discuss features that potentially could be incorporated within this system model to improve user interface.

VI.B.1. Facilitating Data Input. In the Smart Graphic Environment the designer supplies the geometric object data of the digital heritage building record primarily by using the various drawing tools supplied within a CAD environment (see also Section V.E. - Creating Digital Heritage Building Records). Designated layers are used to view photographic (raster) images. Special windows or *palettes* are used to enter non-geometric attribute information.

It is worth noting that multiple viewing interface features, as found in (Drach, et al. 1993), have been successfully used to enter and control data in other KBS design systems. In these environments the approach is to provide a geometric and conceptual of view of the same (proposed) design information as well as representations of the (required) design constraint models used by the knowledge base to evaluate the design information. This includes a geometric view, association view (semantic networks), text windows (attributes), prototypes (view of object constraints used by the knowledge base) and prototype editors (facilities for turning on and off certain constraints from within the CAD environment) (Figure VI.2.). Providing straightforward means of interfacing with the knowledge base object constraints makes it easier to update and maintain the knowledge in the system.

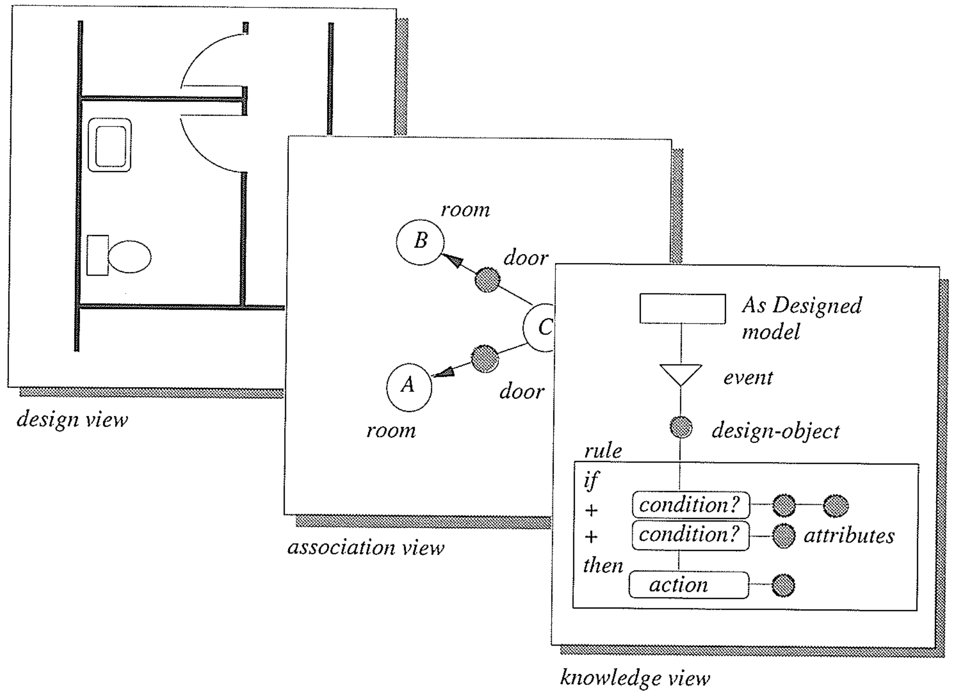


FIGURE VI.2. - Design, association, and Knowledge Base views of the same design information (after Drach, et al. 1993)

An additional feature that is helpful is one that instructs or queries the designer for missing or incomplete data that is needed for particular types of evaluations. This accommodates the needs of the novice system users while at the same time maintaining functionality for more experienced users.

VI.B.2. Requesting & Presenting Feedback. In the Smart Graphic Environment the designer invokes interfaces to the database and critiquing through the CAD environment. Special programs, called *macros,* created within the CAD system allow it to communicate with the database† and KBS. This means that additional commands are added to the CAD system. The Designer is able to request an analysis report for a given design by simply selecting the appropriate command from the CAD system's menus. This will tell the system to update the database records associated with the current design (if modifications were made) and then invoke the KBS. The KBS loads the appropriate database records, apply its knowledge and returns a text file containing the analysis report. This report is then displayed within the CAD system in a *conflicts window*. This process continues at the designer's discretion.

The analysis report should present information to the user in a manner that is appropriate to make ideas clear. Business people often use pie charts or other kinds of graphics to make statistical data easier to understand. In fact using more than one technique for presenting the same information is an effective way to communicate ideas to people. Presenting report information in different formats, such as a spreadsheet or graphic image(s) (examples), so that it can be exported and manipulated by other applications is desirable.

A simple approach to enhance the reporting feature of the Smart Graphic Environment can be to simulate the red-lining procedure. This is accomplished by building in special procedures that enable the KBS to send not only an analysis report, but also instructions to the CAD system to highlight design objects referred to in the report. These procedures instruct the CAD system to select objects and changing their display colour.

Ultimately, the reporting feature of the Smart Graphic's environment should be explored in depth and customized for each type of domain analysis that is provided. Trial testing with potential user communities is the best approach for determining the

† Some commercial CAD systems (i.e. MiniCAD+, Microstation Mac, ClarisCAD) already have connections to spreadsheet and/or database programs imbedded in them.

effectiveness of any interface. Chapter VII provides examples of the Smart Graphic Environment interface developed for prototype systems that analyze building code compliance of heritage buildings.

VI.C. References

Aikin, O. 1988. "Expertise of the architect." *Expert Systems for Engineering Design.* Academic Press, Inc. 173-198.

Archea, J. 1986. "Puzzle-making: What architects do when no one is looking." in *ACADIA '86 Proceedings - 'The Computability of Design.* Buffalo: State University of New York.

Browning, J. 1992. "Artificial Intelligence - cogito, ergo something." *The Economist.* 14 March 1992. B. Cleaton (ed.) New York: Newspaper Group Inc. 5 - 22.

Coyne, R. 1990. "Logic of design actions." *Knowledge Based Systems.* 3 No. 4. Sydney: Butterworth-Heinemann Ltd. 242-257.

Drach, A., M. Landenegger, S. Heitz. 1993. "Working with prototypes, from CAD to flexible tools for integrated building design." *Design & Decision Support Systems in Architecture.* H.Timmermans (ed.). The Netherlands: Kluwer Academic Publishers. 61-82.

Mitchell, W.J. 1990a. "A new agenda for computer aided design." *Electronic Design Studio.* M. McCullough, W.J. Mitchell, and P. Purcell (eds). Cambridge: MIT Press. 1-15.

Mitchell, W.J. 1990b. *The Logic of Architecture - Design, Computation, and Cognition.* Cambridge: MIT Press.

Shneiderman, B. 1989. "Designing the user interface." in *Computers in the Human Context.* T. Forester (ed.). Cambridge, MA: MIT Press. 166-173.

CHAPTER VII

IMPLEMENTATION

This chapter describes the implementation of two prototype KBS capable of reviewing an historical building design for conformity with the 1990 version of the National Building Code of Canada [NBCC]. This part of the research was undertaken with the assistance of the Department of Computer Science at the University of Manitoba and the Plan Examination Branch of the City of Winnipeg.

The prototypes were developed to test the conceptual framework developed for the Smart Graphic Environment presented earlier in this thesis. This work also builds on the existing research in the application of expert systems in automated building code compliance checking. A comprehensive bibliography of this field of research is found in (Vanier 1991b, 250). The first prototype utilizes a commercially available expert system shell with a data sheet interface. The second prototype was developed using an object-based approach to represent and integrate both building data in a CAD environment and domain expertise in a KBS. The object-based approach facilitated the exchange of data between the designer, the CAD environment, and the expert system. The object-based knowledge representation scheme also improved the process of developing and maintaining the knowledge base in the system.

The prototypes address NBCC requirements which are relevant to accommodating persons with disabilities in an existing structure. The requirements for barrier-free design include many of the types of code requirements found in regulatory documents. A subset of requirements for public washrooms, known as "special washrooms," was chosen as a particular focus for the prototypes. Special washrooms are commonly used by designers to replace or augment washroom facilities in historic buildings that were not designed to accommodate the persons with disabilities. These washrooms are unisex facilities containing at least a toilet with grab bars, sink, accessible entrance, mirror, and maneuvering space for wheelchairs.

Choosing to focus on a special washroom design problem meant that the prototypes could demonstrate the potential of the Smart Graphics approach in an area of interest for both conservationists and building code officials, while at the same time keeping the project to a manageable size. Furthermore, much of the expertise in this area was available through the team members who contributed to this phase of the study.† Utilizing readily available expertise makes problem analysis and knowledge acquisition more straightforward than extracting this information from external consultants. Similarly, Knowledge Base System programming is best left to persons with a computer science background rather than domain experts.

The scope of the prototype is limited to the analysis using:

(1) functional,

(2) object based, and

(3) contextual constraints,

of the NBCC for special washrooms. Functional constraints are requirements that deal with the necessity of the presence of particular objects in the room. For example, a

† The author is Director of a research institute, Canadian Institute for Barrier-free Design, specializing in the design needs of persons with disabilities. **John Frye**, Chief Plan Examiner for the City of Winnipeg and developer of ACCEX a similar but non-CAD integrated expert system, provided additional domain expertise. expert systems specialist, **Dr. Mark Evans** and doctoral student **Linda Strachan,** both with the department of Computer Science at the University of Manitoba, helped to determine an effective application development strategy and undertook the necessary computer programming for the prototypes.

washroom must have a toilet object. Object-based constraints deal with the desired characteristics of the objects themselves, such as critical dimensions or shape. Contextual constraints deal with the issues of relationship and orientation between objects such as clearances or minimum distances. These requirements impact on decisions about physical planning and removal or alteration of existing conditions.

The following sections describe the methodology used to create prototypes I and II. This includes a description of the representation of both building data and expert knowledge, and control strategies. The benefits and shortcomings of the resulting design decision-making support environment are discussed at the end of the chapter.

VII.A. Implementation Issues and Objectives

As mentioned in the preceding overview, the purpose for the creation of a working prototype is to test the concept of providing design decision-making support for the redevelopment process using a Smart Graphic Environment. There is much to be learned through the implementation process about the practicality of such environments. Not only in terms of what it takes to develop a knowledge base system, but if such a system can be a useful tool for the designer.

The prototype, to be an effective design decision-making support system, must demonstrate four characteristics:

(1) utility,
(2) flexibility,

(3) use available technology, and

(4) be easy to use.

To be useful, the prototypical Smart Graphic Environment must have the abilities and tools to perform the tasks a designer wants it to do, communicate ideas graphically (as in a CAD environment), and deliver results quickly. Flexibility indicates that the design environment is usable in a variety of ways, at different times, with a good level of control by the designer. Not every designer uses the same design process. Using commercially available hardware and software to implement the system means that it would be more affordable and easier to maintain than a totally unique one. Of the four issues, however, user interface for the prototype is considered to be by far the most important consideration. Even the most powerful system in the world will be abandoned if it is too difficult to learn and use. As in all things, the simpler the prototype was, the better.

To date, the most effective KBS that have been implemented are narrow in scope, but deep in understanding. They are very specialized in a given domain. There is a limit to the number of rules they can handle. They do not learn from their mistakes, so they are difficult to maintain. Most collect information from the user, usually in the form of questions, although they can collect information from a database. KBS are very good at identifying problems, but not as good at innovatively solving them. The concept behind the Smart Graphic approach for the prototype development is that, as a design decision-making support tool, knowledge systems are best at providing critical evaluation within a particular domain. It should be possible, therefore, to create a design environment that combines expert critiquing power of a knowledge base with the creativity and decision-making capabilities of the designer. Since decision-making is an on-going process during design, critical evaluation has to be available at any time. This means that the KBS should be accessible within the CAD environment itself. Depending on the evaluation, the KBS should be capable of finding the information it needs to provide critical evaluation contained within the building design CAD model and IGDB.

With the understanding that designers that use computers generally utilize commercial software and microcomputers because they are affordable and readily available, the prototype was developed and tested for a microcomputer environment. A commercially available CAD software package, with a fairly standard set of *creation tools*, was used as the design environment. It would be from within this environment that a connection between the designer and the KBS would be created.

The long term objective of this study would be to define knowledge bases that could evaluate a wide variety of historical buildings and their possible component objects in relation to redevelopment issues. Given the time constraints of the development period, four months, the scope of the prototype had to be limited, while at the same time reveal the validity of the approach for the development of a more comprehensive application. Thus, a single evaluative task within a relevant domain was chosen - the evaluation of the building code compliance of a room within a digital heritage record. This task is both relevant to the area of interest, the redevelopment of heritage buildings, and has an accessible area of expertise.

The objectives of the prototype Smart Graphic Environment are to:

- **evaluate** (identify problems and applicable building code violations)
- **explain** (why the problem exists and what the correct course of action should be)
- **educate** (to illustrate or give examples).

Creating a KBS that evaluates building code compliance requires the acquisition of knowledge about the building code constraints and how they are applied. In this case, a protocol for the evaluation must be established that also takes into consideration the value of historical elements. Once this expert knowledge is collected it has to be represented and structured in such a manner that describes how the knowledge is used. This representation is then used by the programmers to create the actual computer code of a KBS (Figure VII.1.).

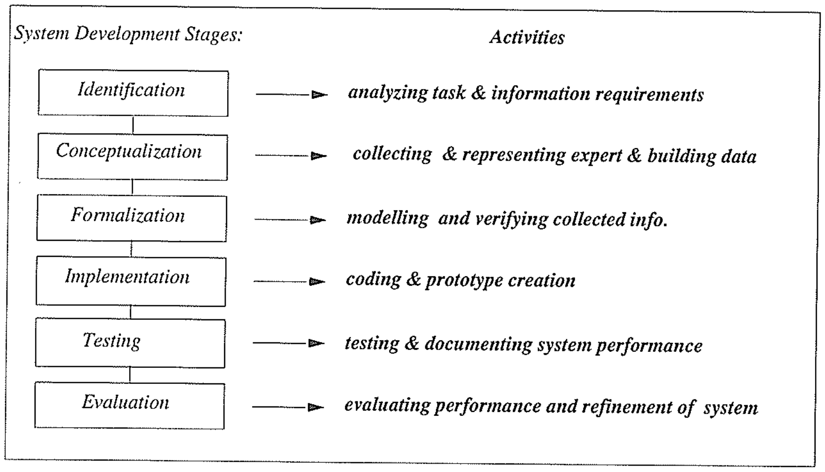

System Development Stages:
Activities
Identification
analyzing task & information requirements
Conceptualization
collecting & representing expert & building data
Formalization
modelling and verifying collected info.
Implementation
coding & prototype creation
Testing
testing & documenting system performance
Evaluation
evaluating performance and refinement of system

FIGURE VII.1. - SGE system development stages and activities

Knowledge about the application of building codes to heritage buildings for this study was gleaned from code documents, related research articles, and personal interviews. The techniques used for knowledge and building design representation were based on recent research in the area of building code automation and input from the team's programmers. For Prototype I, the KBS was created using a commercial expert system shell called *LEVEL5 Object*. In Prototype II, the KBS was created using a programming language, Macintosh *LISP*. Building information was modeled in a commercial CAD program, *MiniCAD+*.

The following sections discuss in more detail the knowledge acquisition and structuring techniques described above and the resulting prototypes.

VII.B. Regulatory Requirements and Heritage Buildings

A consequence of redevelopment of a heritage building is that an owner will be required to use modern building codes. This can have a dramatic effect on the potential occupancy and cost of the renovation. Achieving compliance of historical buildings with modern building codes can be difficult. Building codes are constantly upgraded to respond to new materials, construction practices, and the lobbying of interest groups. Any evaluation is almost guarantied to reveal deficiencies in an older building.

It is very important that code conflicts be identified early to avoid costly delays or revisions. Most preliminary code reviews are performed by the architect during the design process. However, identification of code requirements can be problematic. Regulatory documents contain thousands of detailed requirements that vary in their application depending on the building and its intended use. Although this information is relatively stable, few architects spend the time to become familiar with them. The task of manually researching voluminous regulatory documents is considered by most architects to be an onerous one. This review process is further complicated when it involves a heritage building.

Many heritage buildings do not fall into building categories defined in modern codes. Details such as fire ratings for archaic materials and assemblies are not given. As a result many architects will seek out a specialist to review their redevelopment designs for them. The scarcity of information and need to rely on a external consultants to provide expertise are common reasons for utilizing knowledge systems. By encapsulating uncommon knowledge and making it available to designers on their own desktop represents definite savings in terms of cost and time.

VII.C. Knowledge Acquisition - Establishing a Protocol

The first stage of the prototype development required establishing a protocol for applying the NBCC to heritage buildings in general and then, more specifically, to special washrooms. The following section contains a description of the procedures used for acquiring knowledge from review of code documents, relevant research articles, personal consulting experience, and interviews with a domain expert.

VII.C.1. Building codes. In North America the responsibility for building regulation rests with the state, province, or territory. In general, this responsibility is delegated to municipalities. In order to limit the variations between municipalities, model codes are adopted. Most states in the U.S. have adopted one of three model codes, one produced by the Building Owners & Code Administers [BOCA]. The National Building Code of Canada [NBCC] is a model code that is adopted as law in most provinces in Canada with some modification for regional or geographical differences.

Building code requirements generally represent one of two types of approaches to compliance, prescriptive or performance. Prescriptive constraints specify acceptable materials, sizes or construction methods. The NBCC is primarily a performance based code. It recommends a minimum acceptable standard for the construction material or system rather than particular materials or construction techniques. This allows the flexibility for official authorities to exercise judgment in its application. The NBCC contains approximately 3000 requirements relating to life safety, property protection and structural integrity. Related documents to the NBCC that provide technical and explanatory material include: the *National Fire Code of Canada* (NFCC 1990), the *Canadian Housing Code* (CHC 1990), and *Supplement to the National Building Code* (CCBFC 1990).

VII.C.2. Application of code requirements to heritage buildings. Determining whether or not the building code applies to a heritage building is dependent on *triggering.* Triggering is a term that refers to an event that mandates the application of new construction code requirements onto a rehabilitation project. There are four triggering mechanisms for the NBCC: (1) renovation or alteration (2) change in major occupancy, (3) change in owner (4) retroactive or maintenance laws. Unless one of these conditions apply, achieving compliance with the most recent version of the NBCC is not required. Of course, in the case of the redevelopment of heritage buildings, it can be assumed that at least one or more of these conditions is certain to exist. Depending on the triggering that occurs all or only part of a heritage building may require upgrading. This issue is described in more detail in Appendix A and would have to be considered in a more comprehensive system.

The application of the different sections and articles contained within the NBCC is a function of building *classification.* Buildings, or the parts of a building, are classified according to occupancy, construction type, materials, height, siting and area. This is because health and life safety requirements are dependent on proposed use. For example, the egress requirements will be more stringent for an area used by many of people, like a theatre, than for an unoccupied area. Similarly, health requirements, such as the number and location of special washrooms, are also dependent on 'global' building information.

During an interview, the domain expert explained that, over time, human plan examiners have developed their own system of exclusion and code explication based on experience and understanding of code structure. A building code expert will use their knowledge to exclude non-applicable constraints during a review. They tend to follow a particular assessment path from large issues to more detailed requirements based on the building size and occupancy. Once the applicable rules are identified the proposed design is checked against them. It is only after problems are identified with the proposed design that the examiner and the architect begin to consider alternatives in light of the building or the objects within the building having historical significance (Frye, Personal interview 1991).

In any redevelopment project the maintenance of historical integrity must be weighed carefully against the health and life safety requirements established by regulatory codes. Regulatory bodies have recognized that heritage buildings represent a particular problem for which some compromise in the goals of health and life safety may be required (Hattis 1981a, 11). In general, the resolution of conflicts involves negotiation with local building code officials. An owner who is having difficulty with compliance will be referred to other building codes or guidelines

In some cases model codes have attempted to address the needs of existing structures. In 1984, BOCA created the *National Existing Structures Code* as part of their Basic/National Codes. Some areas in the U.S. utilize special review procedures to respond to the needs of historic buildings, as described in (Hattis 1981a). In 1993, the Canadian Commission on Building and Fire Codes published *Guidelines for Application of Part 3 of the National Building Code of Canada to Existing Buildings*. Guidelines are not codes, but are intended to be used to help with the application of modern building codes in existing buildings, including the renovation and restoration of historic buildings, through clarification of intent and examples.

Guidelines tend to focus on issues such as the cost-benefit factor -- when the costs of upgrading severely outweigh the achieved benefits. In heritage buildings one of the "costs" may be the loss of an essential historic feature that is not easy to quantify (Hansen 1984, 230-2). The authority having jurisdiction has the discretion of weighing these factors. In some cases the owner may be requested to get the opinion of a specialist or involve the representatives of the disabled or heritage community in resolving design problems.

VII.C.3. Accessible Heritage. The most common code conflicts of existing historical structures fall into one of three categories: structural, fire and life safety, and accessibility to the disabled (Prudon 1990, 31). Whereas structural requirements include both large and small buildings, performance requirements relating to fire safety have a greater impact on large historical buildings because the fire can be large and escape may not be readily achievable from upper storeys. Accessibility requirements are usually most difficult to achieve in smaller heritage buildings because of spatial limitations. Ramps and washroom facilities for persons with disabilities can take up a considerable amount of space and can impact heavily on the aesthetic appearance or configuration of plan.

Providing access to historic buildings for persons with disabilities is discussed in (Prudon and Dalton 1981), (Parrott 1980), (Ballantyne 1983) and (Smith and Frier 1990). It was also the subject of a series of conferences entitled *Accessibility and Historic Preservation* held in 1992 and 1993 (Jester and Hayward 1993). Based on building traditions evolved from symbolic or ceremonial functions, most heritage buildings fail to accommodate the needs of persons with disabilities (Parrott 1980). In recent years, the combination of an aging population and the evolution of assistive technologies has produced a greater integration of persons with disabilities into the community. The exclusion of the disabled population from buildings and services is no longer an acceptable practice.

The incorporation of standards for barrier-free design in Subsection 3.7 of the NBCC reflected an all-encompassing effort to remove any and all barriers from the environment and to create spaces responsive to the broadest spectrum of the population. It is within this section that the specific requirements for special washrooms are found.

VII.C.4. Special washrooms. In general, building code constraints relating to barrier-free design are directed at correcting either dimensional inadequacies (door widths, clear spaces) or failure to provide assistive devices (signage, grab bars). Given an example of a special washroom, the domain expert† identified the following process evaluation based on the NBCC:

1. Determine the occupancy and occupancy load of the building design;
2. Check to see if the number of lavatories provided in the building design exceeds the number required based on Article 3.6.4.1. (Table 3.6.4.A. - C);
3. Apply Subsection 3.7 to the design; Articles 3.7.2.3. ("Washrooms Required to be Barrier-Free"), 3.7.2.3. ("Signage"), and 3.7.3.8. - 11. ("Water Closet Stalls," "Water Closets," "Lavatories," and "Special Washrooms"); and
4. If the building is existing consider mitigating circumstances based on guidelines for washrooms in (CCBFC 1993, 37) and other barrier-free design guidelines such as (CPA 1989) and (CSA 1990).

† John Frye, Chief Plan Examiner, Department of Plan Examination, City of Winnipeg.

A specific example of this process would be: a room is identified as the only special washroom located in floor area of a converted warehouse building with a commercial occupancy. Based on this knowledge, a code expert would select Article 3.7.3.11. of the NBCC, which specifies the requirements for special washrooms, as being applicable. The next step would then be to attempt to verify the compliance of the washroom in question by checking the attributes of the design against the code requirements. If a conflict with the code is identified the impact of resolving the problem on historic value has to be considered. If, for example, there is insufficient clear opening underneath a lavatory with legs or a deep apron to permit a person in a wheelchair direct access to the fixtures the designer may have to remove and replace it with a wall hung lavatory. This would allow independent access and increase wheelchair maneuvering space. However, one would also have to consider if making this change altered the historic appearance or damaged the historic fabric. Patching the floor and wall to match existing materials may also be difficult (Ballantyne 1983, 42).

VII.C.5. Establishing a protocol. Based on interviews with the domain expert in this study and a hybridization of protocols described in (Markman 1981, 19) and (Dym, et al. 1988, 138) (see Appendix B), applying building codes to heritage buildings can be summarized in the following steps:

1. identify existing building elements and construction,
2. calculate of level of performance & classification,
3. identify applicable requirements of the NBCC,
4. compare of NBCC requirements to given design (identify conflicts), and
5. consideration of contingencies.

The building code describes a goal state or ideal model (As-Required), whereas a digital heritage building model describes existing building elements as they are (As-Built or As-Designed) (de Waard and Tolman, 1991). What the automation of the building code review process involves, then, is the creation and comparison two models, one of the building design and one of the building code constraints. All building code constraints contained in the NBCC do not apply to every building. A public restaurant has different requirements than a private residence. The attribute values of a building's elements (i.e. occupancy and size) are identified so that the building's NBCC classification can be established. This identifies an applicable subset of all NBCC requirements relevant to the particular building instance being evaluated (Figure VII.2.). Problems are identified and reported to the user by comparing the performance values of the building model and NBCC constraint model. Certain contingencies, such as the historic value of a particular element or the structural performance of a particular archaic material (equivalencies), may also need to be considered. The KBS system can assist in the resolution of the non-conformance problems it finds by providing further clarification of the building code intentions and examples in the same way a human building authority would.

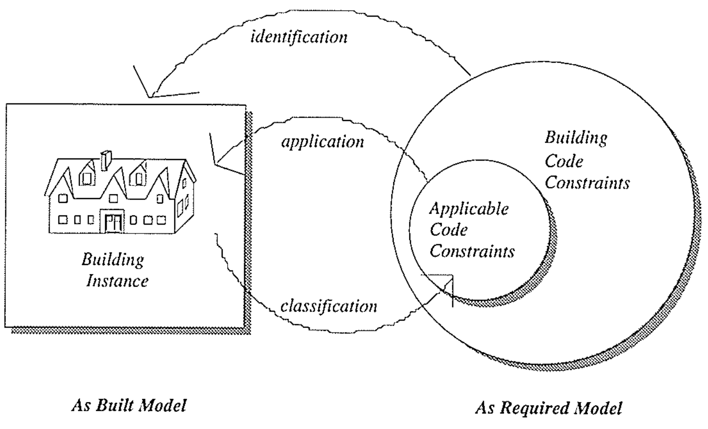


FIGURE VII.2. - Relationship between protocol and building and constraint models

VII.C.6. Specific selection and source of prototype constraints. For the most part, the specific constraints to be modeled in the knowledge base of the prototype are from Subsection 3.7 "Barrier-free Design" of the 1990 version of the NBCC. Sentence 3.7.1. is used to establish exclusion rules in the knowledge base for building classifications that do not need to comply with this subsection of the Code. Since there are only a few building types exempted (i.e. pumphouses, service and detention areas) it is easy for the system to check for them. The majority of other constraints used in the prototype are based on Article 3.7.3.11 *Special Washrooms.*

The constraints fall into one of three categories:

(1) functional,

(2) object-based, and

(3) contextual

Functional constraints identified required objects in the room to satisfy the general category of "special washroom." These constraints deal with the necessity of the presence of particular objects in the room (i.e. toilet, sink, grab bar, . . . etc.). Object-based constraints deal with the desired characteristics of the objects themselves such as dimensions or shape (other examples include: the type of lock, door hardware, door width, faucet type, loading capacity of grab bar, . . . etc.). Contextual constraints deal with the issues of relationship and orientation between objects such as clearances or minimum distances. Resolution of any these three types of constraints can impact on decisions about physical planning and removal or alteration of existing conditions.

Additional constraints were based on the requirements of "washrooms," "accessible washrooms" and "special washrooms" as described in the ACCESS Manual, and the CAN/CSA-B651-M90, *Barrier-free Design - A National Standard for Canada.* Additional expertise for the resolution of problems specific to heritage buildings was taken from (Ballantyne 1983a & b).

VII.D. Knowledge Representation - Conceptual Modeling

The purpose of the following Section VII.D. is twofold. Firstly, to summarize a portion of the recent research that is relevant to the development of the automated building code compliance evaluation systems. Secondly, to describe the conceptual modeling techniques used to capture expertise required for the prototype.

To date, no computerized code compliance checker deals specifically with the problem of applying building codes to existing or, more specifically, historic buildings. However, it is possible to relate techniques for the automation of building codes to techniques for the application of building codes to heritage buildings to establish a basic framework for the development of a prototype KBS.

One of the most important stages before implementation of the prototypes is the creation of conceptual models of building code and building design data. Conceptual modeling is a technique used to document collected design knowledge and data relationships (de Gelder and Lucardie 1994) (Turner, 1991) (Di Battista, et al. 1989) . The most important advantages to conceptual models are that: (1) the models are easier to work with than final data structures since they are developed independent of any type of efficiency and storage considerations related to a specific application, (2) the information remains more stable and can be used in different environments, and, (3) the models are easier to understand by non-experts or non-programmers and therefore are easier to validate (de Gelder and Lucardie 1994, 6).

<u>**VII.D.1. Expert systems and building code compliance evaluation.**</u> The ever-increasing number and complexity of building code requirements inevitably points to the use of the computer. Research concerning the computerization of building regulations began in the 1970's and has encompassed a wide variety of methodological approaches described in roughly a score of published reports and articles (Björk and Kähkönen 1991, 3). Most of the work to date has been theoretical in nature, however, in the 1990's the first commercial database systems became available to provide sophisticated cross-referencing, glossaries, and indexes for codes as well as building product information (see IRC 1993).

The use of KBS with regulatory codes began in the mid 1980's and has resulted in a number of prototypes (Mitusch 1991) (Taremaë 1991) (Tuominen 1991) and (Frye, et al. 1992). These expert systems represent small samples of national building codes, often dealing with fire regulations, which are well suited for rule-based representation (Björk and Kähkönen 1991, 5). Unlike simple information retrieval or browsing systems, these expert systems are capable of intelligent searches, dealing with "what if" questions, and can weigh alternative design solutions.

In the 1990's, research in this domain is progressing towards the integration of automated building codes within CAD systems. The principal challenge in this area has been to create common standardized data models for representing both building codes and building descriptions. Of course, building codes were developed well before computing tools and techniques were a consideration. Data exchange is a common problem, even between CAD programs. Information required for a code checking procedure can be found within the CAD model only if one bridges the gap between the way industry defines buildings and the definitions established by building regulations.

In some cases, the study of integrating knowledge technology and technical regulations has been directed towards assisting in the development and management of such regulations. This has led to the development of schemas and classification of the general concepts contained within regulations (Lucardie 1993) (Vanier 1991). A rationalization of the goals of the regulations provides better understanding of the regulations themselves. Having accurate knowledge allows one to choose the right actions to achieve a goal. This is what is known as a *knowledge level* approach.

Knowledge level models, based on a goal-oriented classification of knowledge and data, accommodate a more flexible and dynamic approach over simply representing each prescriptive requirement in terms of a rule. The concept of *functional equivalence* described in the conceptual models in (Lucardie 1992, 11) The models show how two very different, but functionally equivalent objects (have attributes that perform the same function), can satisfy the same goal. This is particularly important with regards to achieving regulatory compliance in historical buildings.

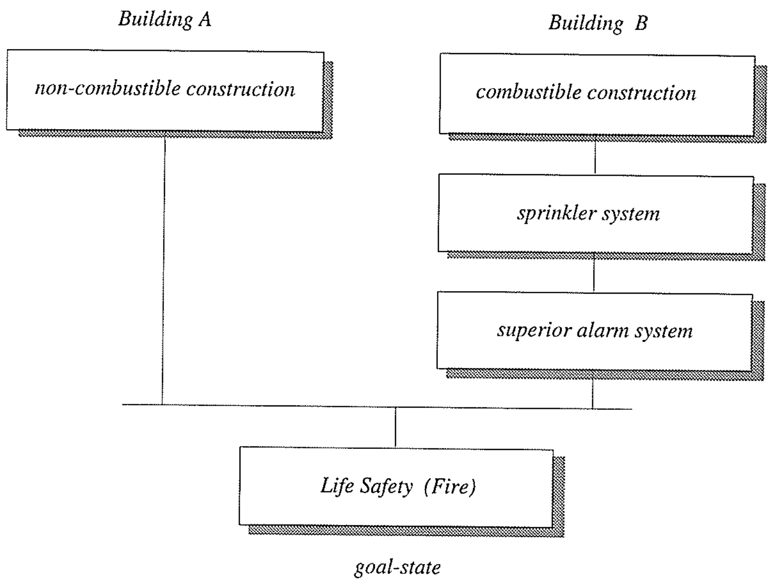


FIGURE VII.3. - Example of Functional Equivalency of Combustible and Non-Combustible Construction in Heritage Buildings

To illustrate the relevance of being able to model functional equivalence we can use an example illustrated in Figure VII.3. A common problem encountered with heritage buildings is that they are often constructed from combustible materials (such as wood) and so the building does not comply with fire safety regulations. A building can be made to achieve code compliance by adding a sprinkler system (fire suppression), and an effective alarm system (early warning). These measures help to achieve the same goal, that of allowing enough time for persons inside the building to escape in a fire, that is achieved in a building of non-combustible construction.

Conceptual models of building codes can show how objects are classified and how attributes can be shared between objects. The work of (Vanier 1991a) is a particularly useful starting point for developing a limited knowledge level model for this thesis study since it develops a schema, concepts, and vocabulary for the NBCC, the regulation used in the prototypes. What Vanier's work identifies is the strong hierarchical nature of information contained in the NBCC. A "special washroom," for example, is a Child Concept of the Parent Concept "washroom." The Child concept can inherit properties or relationships held by the Parent concept as designated by the user. For example, both a special washroom and washroom will contain a "toilet."

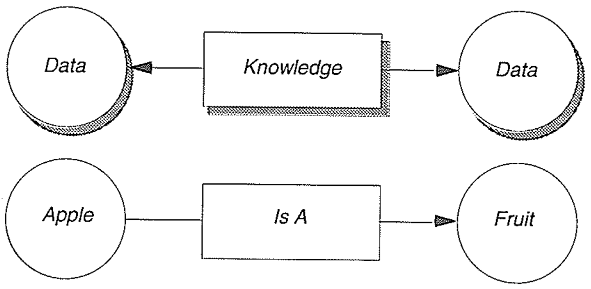


FIGURE VII.4. - Data connected by knowledge

Other relevant information for the purposes of this study comes from researchers who have also adopted an object-oriented approach, including (Cornick, et al. 1991) (Hosking, et al. 1991) (Turk 1991) and (De Waard and Tolman, 1991). In these approaches most of the information needed for the code checking procedure is found in an object-oriented building description, which is the result of the computer aided design process (Björk and Kähkönen 1991, 6) Object-oriented approaches to the modeling of data and knowledge have led to a closer connection of knowledge with the data itself. Where the data forms a general model, the knowledge connects the data together (Figure VII.4.).

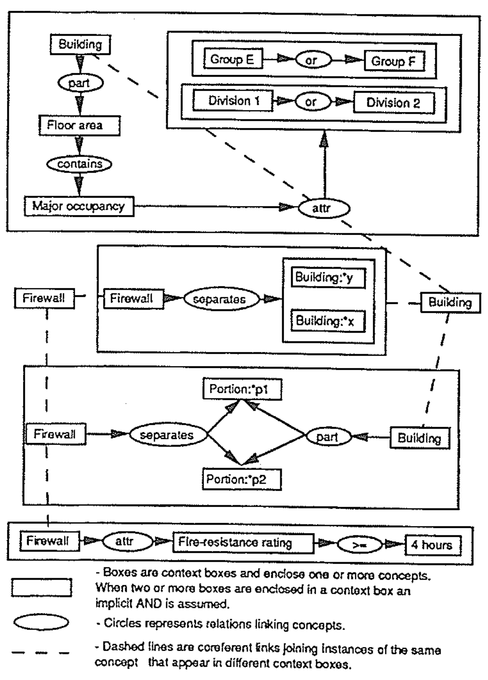

Building
part
Floor area
contains
Major occupancy
attr
Group E
or
Group F
Division 1
or
Division 2
Firewall
Firewall
separates
Building:*y
Building:*x
Building
Portion:*p1
Firewall
separates
part
Building
Portion:*p2
Firewall
attr
Fire-resistance rating
>=
4 hours
- Boxes are context boxes and enclose one or more concepts. When two or more boxes are enclosed in a context box an implicit AND is assumed.
- Circles represents relations linking concepts.
- Dashed lines are coreferent links joining instances of the same concept that appear in different context boxes.

FIGURE VII.5 - A simplified directed graph representing a constraint that a firewall separating a building, or two buildings, containing a Group E or F, Division 1 or 2 floor area shall have a fire resistance rating of 4 hours. From(Cornick, et al. 1991, 225).

Researchers in this area share a common belief that building codes and models of a designed structure can be represented by the same objects, attributes and relationships in a *building product model.* Product models are information models of specific products (De Waard and Tolman 1991). To date very few product models for regulations have been developed. The work of (Cornick, et al. 1991, 210) towards the creation of conceptual reference models for building objects and relationships and the role of constraints in analyzing these models is particularly useful. This work, based on the 1985 NBCC requirements for firewalls, describes a process for generating a *semantic network,* for a design and a constraint in terms of concepts and relations (after Sowa 1984). An example of a directed graph representing a building code constraint is shown in Figure VII.5.

Another approach to product modeling using NIAM† models for building codes is explored in (de Waard and Tolman 1991) and (Turner, et al. 1991). These models recognize a need to create a building product model that accommodates buildings as both spatial and technical systems. In de Waard and Tolman's work additional objects, known as *space boundaries*, are used to connect collections of both space parts (rooms, passageways) with building elements (floors, walls). Object Attributes, attribute values, and procedures (used to determine attribute values) are then overlaid on this model in another process to provide the information required to determine code compliance. The resulting building product model is considered to be the *As-Designed (As-Built)* state and the building regulation, because it describes a building, is similarly modeled as an *As-Required* or goal state. Goal-oriented approaches to product modeling accommodate the idea that it is possible to meet an As-Required state using objects or combination of objects that are functionally equivalent in the As-Designed model.

† Nijssens Information Analysis Method

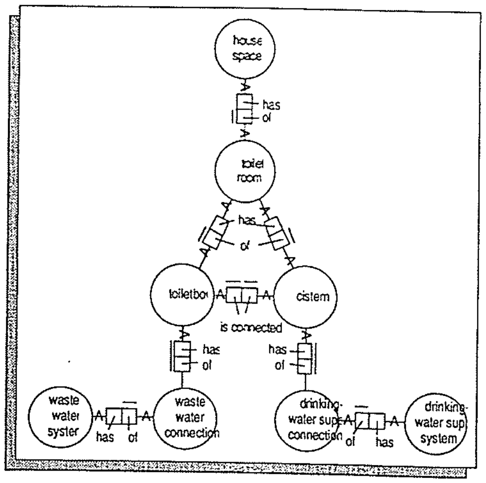


FIGURE VII.6 - NIAM drawing from (de Waard 1991, 198)

<u>**VII.D.2. Summary of representation techniques.**</u> Clearly, the creation of conceptual building product models is a necessary first step in the development of any knowledge system prototype. This approach provides a standard means of visually identifying and structuring objects and relationships (data and knowledge) in a way that is meaningful to an information system. Of course, the more comprehensive the product model is, the more accurate the evaluation can be. Once created, the conceptual building product model (As-Designed state), including historic value, would then have to be accommodated within a CAD drawing and its associated database.

Recognition of functional goal states, used in knowledge level research, provides a more flexible approach for the structuring of knowledge and data contained within building codes. This is particularly relevant to this study since historic structures may achieve compliance by different strategies. Obviously, the ability to present these alternative strategies to the designer is a key factor in the provision of design decision-making support. In order to assist in achieving building code compliance for heritage building a

KBS has to not only understand what the building codes are, and how to apply them to a heritage building but also assist in the resolution of the conflicts that arise.

VII.D.3. Conceptual modeling for the prototype. The conceptual models used for the prototype need only represent a subset of building code constraints and heritage building design information. The only objects that must be in the digital heritage record are the ones required for the evaluation. The identification of significant objects for an evaluation is one of the interesting by-products of the knowledge base development, as they come naturally out of the determination of rules and constraints for code compliance and historical integrity maintenance. Ultimately this fact can impact on the heritage recording process. By recognizing the data requirements of the Knowledge Base for different levels of evaluation one can determine what, when, and how much data should be collected. In other words, it is possible to tailor the collection process to fit the client's needs at different stages of the redevelopment process (Hill, et al. 1992, 4).

Starting with the NBCC building code constraints relating to accessible washrooms, a series of different types of visual aids were created as means of both communicating and structuring the collected data and knowledge to be used in the Prototype. Flow diagrams and/or *decision trees* are used to show decisions and procedures for applying the NBCC used by code experts (Figure VII.7.).

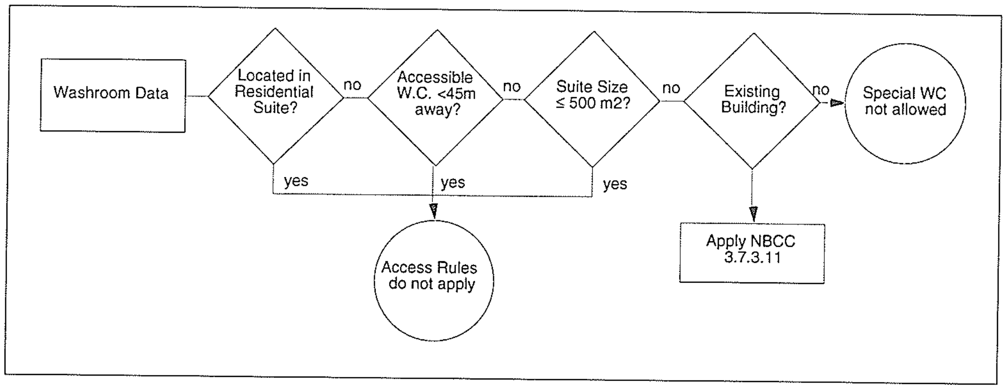


FIGURE VII.7. - Flow diagram showing exclusion rules used in the prototype based on NBCC 3.7.2.3.(2) & (3) "Washrooms Required to be Barrier-Free"

Conceptual graphs are used to show the relationships between objects. The relation represents a link between two data objects. A variety of relations can be used to associate an object with its attributes, values, behaviour, location, parts, and function as described in (Cornick, et al. 1991, 231). Figure VII.8. shows an example of a conceptual graph (based on a simplified NIAM representation) developed for a constraint associated with an accessible washroom door object. Functional equivalence is also described using semantic models as shown in Figure VII.9.

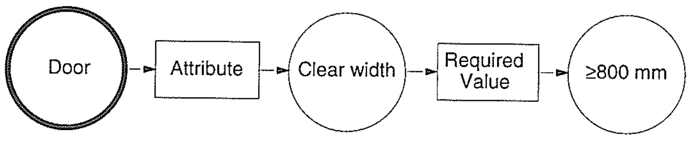


FIGURE VII.8. - Example of door object constraint model

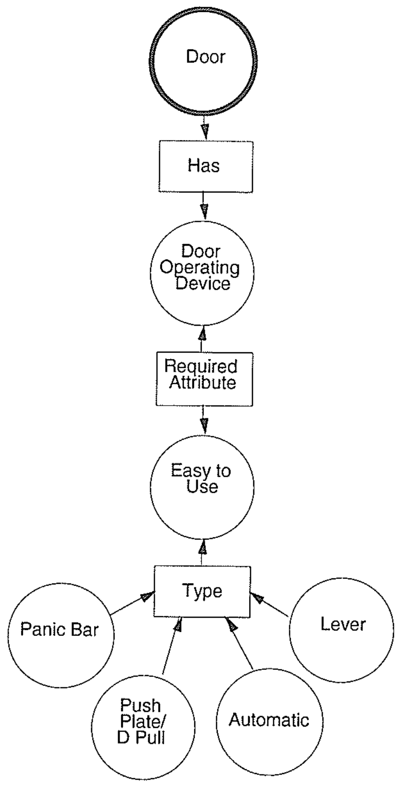


FIGURE VII.9. - Constraint model showing equivalencies of door pull types that meet the intent of NBCC requirement 3.7.3.3.(3) "Doorways and Doors": "Door operating devices shall be of a design which does not require tight grasping and twisting of the wrist as the only means of operation." (NBCC 1990, 131)

Using an object-oriented approach, objects were hierarchically classified in terms of *classes* and *instances*. Object classes represent templates or models of functional constraints, object characteristics, and contextual or object-to-object relationships that are common to groups of objects (Figure VII.10.). A set of procedures and rules is encapsulated in each object to specify this knowledge and how it should be applied in evaluating specific object instances (the object instances are retrieved from the IGDB). Classes are arranged hierarchically so that they can share information. Subclasses inherit the characteristics of their parent classes. Any number of subclasses can be defined and these subclasses can then be refined into their own set of sub-subclasses and so on. Standard defaults are used to enable evaluations when information about the existing building and its component objects is limited or incomplete. This is a powerful mechanism for dealing with partial information (i.e. prototypical information). Defaults can be overridden when actual values for an object instance's attributes become known (Hill, et al. 1993). Eventually the information contained within the hierarchical graph models was transferred into object-oriented frames.

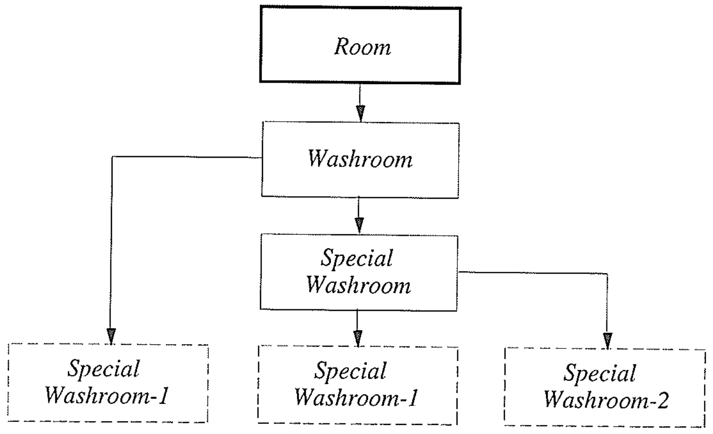


FIGURE VII.10. - Sample object hierarchy

Object-oriented frames were developed for the following object classes: Washroom and Special washroom, and the object subclasses: Door, Wall, Watercloset, and Lavatory. A sample of the object class "Door" is shown in a object class frame in Figure VII.11. Additional object frames created for the prototype are in Appendix C.

Special Washroom DOOR				
SLOT	VALUE(S)	POSSIBLE-VALUES	LEGAL-VALUES	VALUE-TYPE
Knob Type	Lever *	lever, push plate, panic bar, round knob, canted, "D" pull	**ONEOF** (?V, (lever, push plate, panic bar, "D" pull))	LIST
Clear Width	800 *	n/a	(?V >= 800) **OR** (?V >= 760 **AND** Residential)	NUMERIC (mm)
Closing Period	3 *	n/a	?V >= 3	NUMERIC (seconds)

Notes: ?V indicates to access the current value in the slot and use it in the expression. **ONEOF, OR, AND** are built-in functions. These are used to represent specific constraints. " * " denotes a default value.

FIGURE VII.11. - Sample object class frame for the prototype Knowledge Base (Hill, et al. 1992).

VII.E. ENCODING KNOWLEDGE

VII.E.1. Methodology. The process followed by the design team to encode the knowledge and data into a working prototype is summarized in Figure VII.12.

The methodology used allowed the team to benefit from the evaluation of the first prototype before proceeding to the second. The first prototype concentrates on the implementation of the knowledge base system outside of the drawing environment. The purpose of this stage of development was to identify object constraints using conceptual models and frames and then encode them within the knowledge base environment. The second prototype concentrates on the integration of the KBS within the CAD environment, thus creating a Smart Graphic Environment. The objective of Prototype II was to create a smooth exchange of data between the designer and the KBS and CAD systems using an object-based approach coupled with a user-friendly interface.

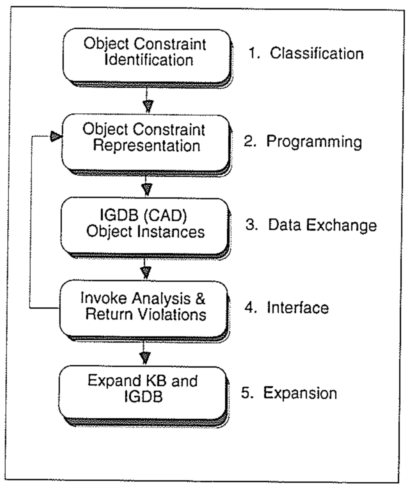

Object Constraint Identification
1. Classification
Object Constraint Representation
2. Programming
IGDB (CAD) Object Instances
3. Data Exchange
Invoke Analysis & Return Violations
4. Interface
Expand KB and IGDB
5. Expansion

FIGURE VII.12. - Methodology (Hill, et al. 1992, 6)

VII.E.2. Prototype 1 - Automating the review process. During the first stage of development a KBS was created outside of the drawing environment using a commercially available expert system shell called LEVEL5 Object. The method of encoding knowledge used in Prototype I is primarily rule-based because of the limitations of the shell in which it was developed. A more powerful combination of a rule-based and object-based approach was used to develop Prototype II.

Rules are the most common type of representation for KBS. Each rule created for the prototype is a simple conditional statement (IF...) consisting of one or more preconditions and a conclusion (THEN...) (Figure VII.13.).

NBCC 1990
Article 3.7.3.9.
(1) Waterclosets for disabled persons shall:
    (a) be equipped with seats located at not less than 400 mm and not more than 460 above the floor.

```
IF    ?SEATHT < 400 THEN ?SEATHT TOO LOW
      ?SEATHT > 460 THEN ?SEATHT TOO HIGH
```

FIGURE VII.13. - Example of code requirement and KBS rules

Prototype I's rule-based system utilizes a *forward-chaining* inference strategy. The inference engine scans, selects, and applies rules and data that apply to a given goal. As the facts about the washroom are collected the inference engine places them in working storage. The rule-base is then scanned to identify violations. This "breadth-first" search strategy (Carrico, et al. 1989), where all of the rules are reviewed proved to be effective because the rule base was not overly large.

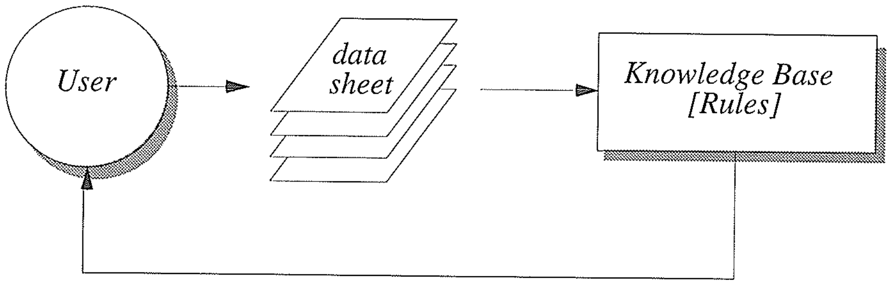
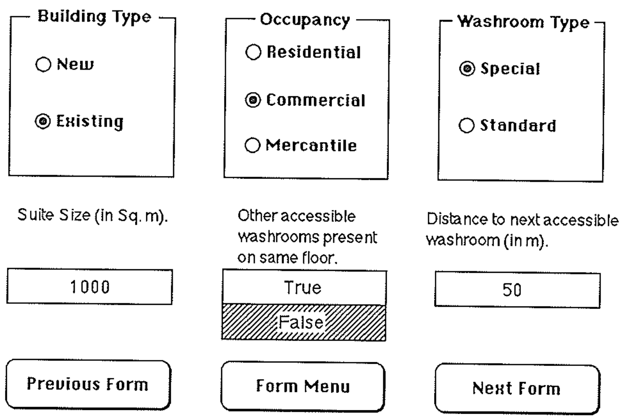


FIGURE VII.14. - Prototype I

At the start of using the system the designer is asked for building classification information to determine whether or not the NBCC applies to their design. Once the knowledge base establishes that analysis is required, forms are presented to the user to supply necessary washroom data. The forms-based input interface facilitated testing of the knowledge base performance. Test washroom data is entered by typing in values and/or point and clicking on predefined parameters. Areas of the form are greyed out or unavailable where evaluation is not possible. For example, if the user indicates that a lavatory is not present within the room then the boxes used to collect detailed information about the lavatory such as faucet type or clearspace are greyed out. The forms are the first representation of object instances that would eventually be read directly from an IGDB of a digital heritage record. Data was edited to simulate design violations. Figure VII.15. illustrates a sample data sheet window used for inputting occupancy information.



Building Type: New [✓ Existing]

Occupancy: Residential [✓ Commercial] Mercantile

Washroom Type: [✓ Special] Standard

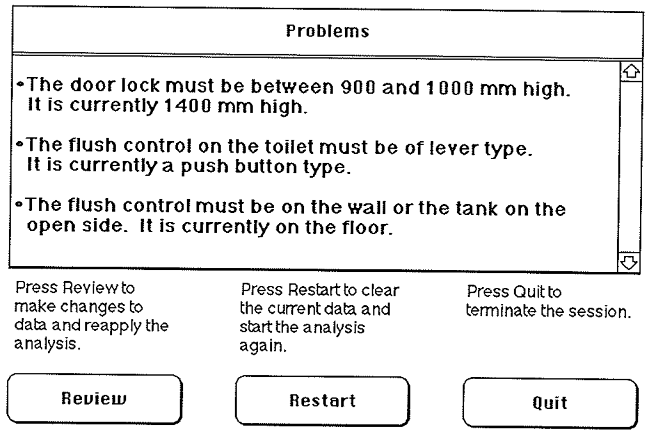
Suite Size (in Sq. m). 1000

Other accessible washrooms present on same floor. True ▨False

Distance to next accessible washroom (in m). 50

FIGURE VII.15. - Sample input data sheet window from Prototype I

The collection process continues until the user invokes the analysis. Analysis is based on the supplied data or defaults. The knowledge base system then returns a specific explanation to describe the problem and the current values of pertinent data. Figure VII.16. shows a sample violation report produced by the system. Validation is provided by applying only rules that reference data items entered or modified by the user. The user is also given the option of reviewing the data before initiating analysis.


Problems

•The door lock must be between 900 and 1000 mm high. It is currently 1400 mm high.

•The flush control on the toilet must be of lever type. It is currently a push button type.

•The flush control must be on the wall or the tank on the open side. It is currently on the floor.

Press Review to make changes to data and reapply the analysis.

Press Restart to clear the current data and start the analysis again.

Press Quit to terminate the session.

FIGURE VII.16. - Conflicts Window from Prototype I

The user is also allowed to make changes to the data and invoke the analysis again. The input forms allowed design characteristics to be easily altered and therefore the performance of the system could be assessed with minimal effort. This feature also supports what-if analysis. An initial design specification can be loaded from the database and evaluated by the system to produce an analysis report. The user can then hypothesize design changes by modifying the data in the forms to see if the changes resolve constraint violations indicated in the report.

This feature created problems where the data changes resolved previously flagged problems. This meant that these problems had to be removed from the expert's conclusions which requires additional rules that recognize valid rules. It became apparent that there was a need to link the data items directly with the problems. Therefore, if a data item was changed then the current information associated with the problems linked to the data item should be cleared.

VII.E.3. Evaluation of Prototype I. The attributes of six different special washrooms were used to test the prototype. The examples included washrooms that met building code requirements and those that did not (Figure VII.17.). Each washroom was analyzed by both a human expert and the KBS and then the results were compared. Walk-throughs of the system with the programmers helped improve user interface and to catch errors. Feedback from initial testing led to modifications in both the content and structure of the knowledge base.

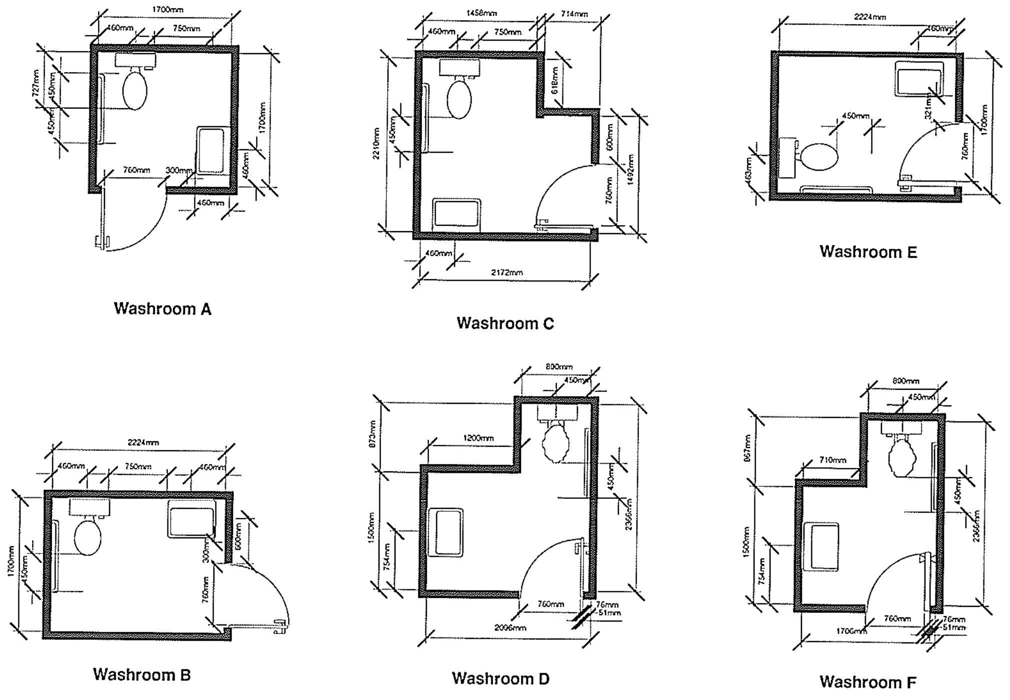


FIGURE VII.17 - Test washroom designs (plan views)

The initial prototype for special washrooms contains fifty-seven rules and was developed and tested in approximately six weeks. The program ran on a typical 386

Personal Computer with a Microsoft Windows™ environment. The investigation was extended to examine whether the KBS could undertake some simple spatial constraint analysis (rectalinear configurations only). This involved adding additional domain knowledge about what objects had to be in the room and their placement and orientation within the room itself. These extensions required twenty additional rules and special purposed code for evaluating object orientations.

Prototype I was remarkably functional, considering the short development period and relatively small rule base. Once the building data had been entered the system returned the correct response(s) almost immediately. The ability of the system to evaluate the washrooms accurately proved the validity of the approach. However, the speed of the evaluation was a function of the size and simplicity of the rule-base. Forward-chaining through a larger, more complex rule base would result in a definite loss of efficiency. This pointed out the need for a more flexible inference strategy than the shell we were using could provide.

As the rules for each of the objects and object relationships were encoded in the knowledge base a list of what information would have to be recorded about the real objects became clear. Therefore, it was shown that the expertise contained in the KBS can be used to direct the data acquisition, helping to limit recording to pertinent information (for code assessment) only. The forms that were used to collect information from the user, because they identified key data objects, created a checklist that could easily be used to save time during on-site recording or inspections.

The data sheet window input, with its multiple choices and input boxes contained within a single topic screen, was a vast improvement over a query-based input. Data was easily modified so that the designer could test a variety of possible scenarios. The inclusion of generic diagrams on the form to clarify what information was being asked for, such as the mid-distance clearance beneath a lavatory or the inter-relationships of object clearspaces, was very helpful even for the experienced code expert. Using forms relies on the user to input all of the building data. This process can be time-consuming and redundant, particularly in light of the fact that much of this information has already been recorded within the design drawings. Therefore, a knowledge base that can read data directly from a CAD file would be more efficient.

The violation reporting used in Prototype I helped to both explain problems and educate the user. For each object that was found to be in violation of the building code, report windows would provide the user with both the object identity, current design value, the required value, as well as brief explanation of the requirement. The capacity to suggest possible solutions in some instances, such as reversing the doorswing to increase the maneuvering space within a washroom area, was also useful.† Although not implemented, it was felt that generic graphic examples of some concepts, such as what is "turning radius" or "easy to grasp handles" should be incorporated in the report as an option.

Our primary research objectives were to create a working prototype in order to test functionality, flexibility, and ease of use of a Smart Graphics Environment. The first prototype, described above, verified the potential of such a system. However, the prototype's flexibility was limited by the use of an expert system shell. Furthermore it failed to allow the user to invoke the expert evaluations from within the CAD environment. The second prototype incorporates an object-based approach and allows for improved flexibility of data exchange between the User, CAD program, and the KBS (Hill, et al. 1992, 6).

† It is interesting to note that a student in Computer Science at U of MB, on his own initiative, used our washroom spatial problem as a test case for a *neural network* system with some success. The system learned to reverse the doorswing automatically in order to increase maneuvering space inside a special washroom. *Automated Design Systems*, or systems that are capable of generating a design entirely by themselves, is an area of great interest to researchers. However these systems, require encoding different types of design knowledge and skills that are outside the scope of this particular thesis.

VII.F. Integrating KBS with Digital Heritage Records: Prototype II - Creating a Smart Graphics Environment

The next step in the implementation process was to integrate the KBS with the IGDB and CAD system. The objective of this stage was to create macros within the CAD system that would allow it to communicate with the database and KBS. Using a "client-server" model, the IGDB and KBS would act as servers to the CAD system. As a result the designer would be able to request an analysis report for a given design by simply selecting the appropriate command from the CAD system's menus. The database records associated with the current design would be updated (if modifications were made) and then the KBS would be invoked. The KBS would load the appropriate database records (object instances), apply its knowledge and return a text file containing an analysis report. The report is then displayed within the CAD system in a conflict window. This process continues at the designers discretion, as illustrated in the schematic diagram in Figure VII.18.

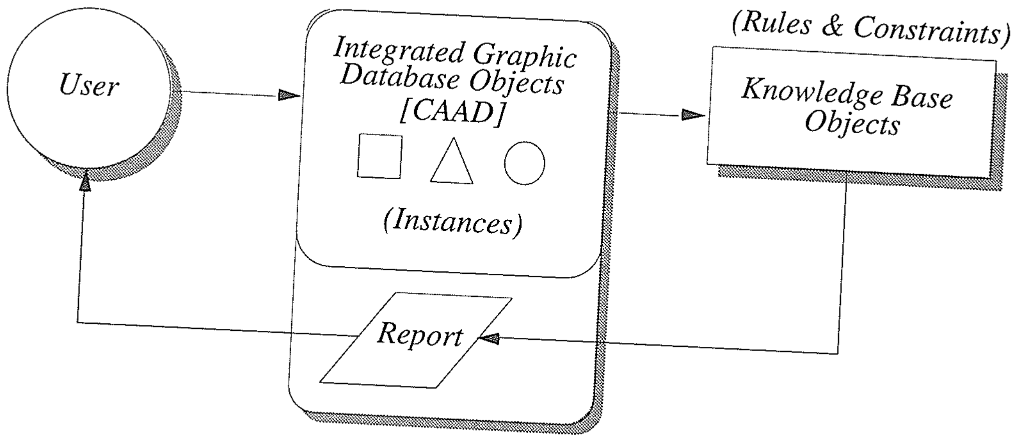


FIGURE VII.18. - Prototype II: Integrated Smart Graphic Environment

VII.F.1. NBCC constraint representation. It was apparent that expert system shell used in Prototype I could not provide the additional representation and inferencing flexibility necessary in order to capture and control the wide variety of knowledge involved in extending the system's scope. Therefore, the special washroom knowledge base was converted from LEVEL5 Object to the programming language LISP. In this conversion the (forward-chaining) rule-based representation was combined with a object-oriented approach. Knowledge, in the form of rules and procedures, was grouped into objects in the Knowledge Base (Hill, et al. 1993).

As described previously, each of the articles from the NBCC relating to the accessible design of special washrooms was represented initially in conceptual frames or constraint models containing object classes and their related requirements. This representation included approximately thirty Articles of the NBCC. Upon completing and verifying these frames, they were then converted by the programmers into computer language code that could be read by the inference engine.

The role of the inference engine in the KBS is to interpret and manipulate the object classes and instances to perform design evaluations. The basic functions of the inference engine are to:

1) extract object instances from the IGDB and create appropriate object instances in the knowledge base.

2) manipulate these object instances and their associated object class knowledge to enforce constraints, explain violations, suggest methods for resolving violations, etc.

In an object-based knowledge base, control is typically distributed throughout the object classes and subclasses defined in the knowledge base. This is done by associating special-purpose procedures (called *methods*) with one or more object classes. Methods interpret and manipulate the contents of the objects that they are contained in. Like data, methods can be inherited from parent classes. This allows general methods for interpreting objects to be shared among many object classes (and hence their subclasses and instances). It also allows specialized methods to be represented directly with objects when necessary. A general control mechanism coordinates the selection of the methods that should be used (it handles inheritance and conflict resolution when necessary). The advantage of this scheme is that the methods for manipulating objects are defined within the objects rather than in a large body of code as is the common approach in procedural computer programming languages such as Pascal and C. Linking the methods of manipulating an object with the object itself makes it much easier to maintain a large knowledge base (Hill, et al. 1993).

VII.F.2. Providing data through a CAD model. Inasmuch as the form-based input of Prototype I was capable of containing object instances required for the knowledge base evaluation, the environment in which architects typically communicate their ideas is with drawings. It is for this reason that a framework for creating Digital Heritage Building Records, described in Chapter III, was developed. Using the conceptual design model and this framework, an object-based two-dimensional CAD drawing and IGDB was created for a special washroom.

A commercially available object-based CAD program, MiniCAD+, was chosen as the environment for developing the digital heritage special washroom record. This CAD system provides standard drawing tools as well as an IGDB. The latter is generally used to generate specifications or quantity survey spreadsheets. For the purposes of the prototype, the CAD building record and IGDB are meant to:

- provide a geometrical description (typical of a standard floor plan),
- represent all of the building objects in a manner that would allow the knowledge base to analyze them using the NBCC, and
- provide a means of efficiently creating and linking objects with their attribute descriptions.

VII.F.3. Collecting, documenting, and modeling heritage washroom data. The heritage recording of a washroom would typically be done by hand measurements. A photographic record is made of unique features such as a washstand sink or decorative tile patterns. Field notes contain other observations such as condition, materials used, or the name of the manufacturer of a fixture if it is identified. Therefore, a heritage record contains unscaled graphics (photographs), scaled or **geometric** information, and textual and tabular, or ***semantic*** descriptions.

It is always desirable to limit the amount of drawing and data input that the user has to do. Photographic records are often used to provide additional information or to create more detailed drawings at a later time if necessary. Referring to photographic images becomes particularly important when modifications affecting the historic fabric are proposed. Photographs can be scanned into the computer and linked to the relevant CAD files. Due to time constraints, however, the integration of raster images, as proposed in the theoretical framework, was not included in the implementation of the working prototypes. The integration of the information found in the raster photographic documentation can be added to the file by the designer by using textual attributes and storing them in the IGDB. The digital heritage building record used for the prototype includes geometric and semantic descriptions directly relevant to the evaluation by the NBCC.

VII.F.4. Geometrical representation. Hand measurements were transferred into the two dimensional CAD environment with the relevant closing errors noted (see Chapter V, Section E). The CAD drawing is made up of objects drawn on different layers according to the standards established by the conceptual framework (see Chapter V, Section F). It is worth noting that the value of layering was immediately recognized and would certainly be necessary in more complicated drawing files. Besides serving a purpose for visualization (the ability to simplify the drawing by turning on and off whether certain objects are shown), the layer designation of particular objects is a further method of assigning a classification. A toilet, for example, would be drawn on the "fixture" (P - FIXT) layer. Layers can also be used to differentiate between new and original construction. The final drawing setup contained twenty-one layers using the same generic divisions created for a more complicated drawing file (as described in earlier in Subsection V.E.3). As a result, not every layer contained information.

CAD drawing tools are used to create geometric entities that represent each of the object instances used in the NBCC evaluation. Objects are used to represent physical elements of the washroom such as the walls and fixtures. Spatial systems are also described with geometric entities representing such things as clear spaces beside fixtures and the room itself.

In the course of creating the geometric drawing file the question of whether or not to use symbols became a consideration. One of the time-saving features of CAD systems is the use of symbols and symbol libraries. In some CAD environment, attributes can only be attached to symbol entities. Symbols are pre-drawn graphic objects that can be placed in a CAD drawing. This is a technique frequently employed for repetitive and standard objects such as toilets. In many cases libraries of symbols are supplied by manufacturers and include additional specifications which can be downloaded directly into the object attribute record.

One of the problems of using symbols for the Smart Graphics Environment is that objects encountered in heritage buildings may no longer be manufactured or be unique in their construction or operation. For the specific requirements of the building code it is important that geometric representation be complete and accurate - so the use of some symbols may be inappropriate. For example the height of the toilet of the symbol object may not reflect accurately enough the actual toilet seat height. On the other hand, aspects of the object which are not required by the code investigation could be represented generically with symbols.

There are two approaches that could be employed for using standard symbols for historical buildings. The first approach is to use symbol libraries created for period style fixtures. Depending on price, availability, and compatibility with the CAD program being used this may be a suitable way of recording existing conditions. The second approach would be to modify a generic symbol in terms of the important attributes required for the building code investigation to reflect actual conditions for each object instance. This may include modification of the graphic representation in terms of its geometrical characteristics as well as the object's attribute record. This would be useful, for example, in a large heritage building that had many instances of the same toilet type. In the case of a single drawing for the prototype it was simpler not to use symbols at all.

VII.F.5. Overlaying semantic information. Since not all of the information required for the NBCC evaluation is geometric in nature, additional attributes were linked to the objects in the test drawing using *Architectural Attribute Coding*. Object instances of the As-Required conceptual constraint model, encoded in the knowledge base, must be identified within the As-Designed model, encoded in the CAD file. Geometric entities in the CAD drawing are assigned Classes and Names. CAD object classes are the same as KBS Classes, such as Room, Door, Toilet, Lavatory and so on. An object's "Name" indicates an Object Instance of a class of object, for example "Door-1" is an instance of the Class "Door."

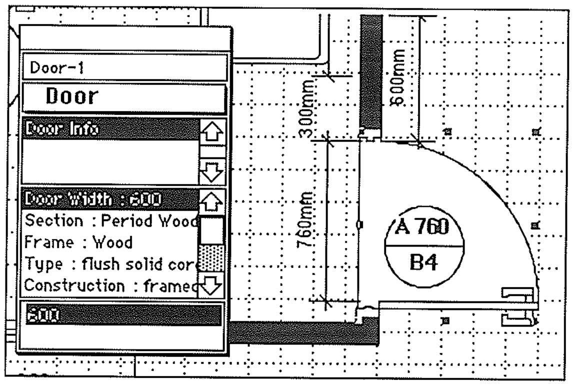
Using the IGDB of the CAD system, records and fields are created to contain object semantic descriptions. Records contain groups of fields. Fields contain attributes and attribute values attached to each object. Predefined records and fields of the AAC are used to save time in linking classes of objects with their shared attributes and to ensure consistency with the KBS definitions. As each object is created the relevant record is attached to it and fields filled out in terms of known attribute values. MiniCAD+™ has a special *data palette* that is used to input field information (Figure VII.19.).

An object's *explicit geometric* attribute values are taken directly from the drawing, such as the object's height or width. *Calculated geometric* attribute values are derived from other attributes using procedures imbedded in the database, such as the calculation of an object's area or volume. Explicit and calculated geometric attribute values are dynamically linked to an object entity in the drawing so that database values are automatically updated to reflect any modifications that are made to an object in the drawing.

Non-geometric attributes are assigned to objects to cover data that is not connected to the geometric information of the object, but is necessary for the analysis. For example, non-geometric characteristics include: (1) *physical attributes* such as the color, type, or material of the door or (2) *semantic attributes* of historic value or condition of a particular wall finish. *Default attributes* † may also be temporarily assigned where information may

† Using a relational database for the SGE would make it more efficient. This means that a number of predefined databases would be used. The object records would contain code that would recall predefined object attribute data held in databases outside of a particular CAD IGDB file. This means that individual file sizes can be considerably smaller. This involves a change at the programming level only as it is done entirely transparent to the user. An example of this is that when a user selects the relevant door type from a list on a pull-down menu in the data palette the appropriate coding (tag) is automatically added to the object

be missing or incomplete but an analysis is desired. These preset values are based on inferences from the rules required for the analysis of the building.


Door-1

Door

Door Info

Door Width : 600

Section : Period Wood

Frame : Wood

Type : flush solid cor

Construction : frame

600

FIGURE VII.19. - Data Palette and washroom door object

<u>VII.F.6. Analyzing the digital heritage washroom design.</u> Once the drawing and the AAC have been attached by the designer, the analysis can be initiated (Figure VII.20.). This procedure updates the IGDB with all the data contained in the CAD building design record. The contents of the IGDB is then downloaded into the KBS as Object Instances. The inference engine compares each Object Instance with the object constraints contained knowledge base. When the analysis is complete a message is passed to the CAD environment and different types of information are presented to the user.

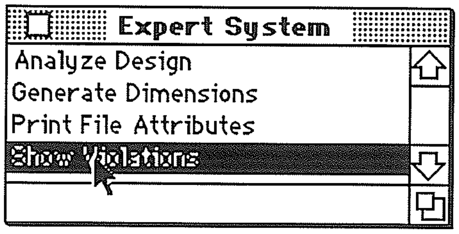
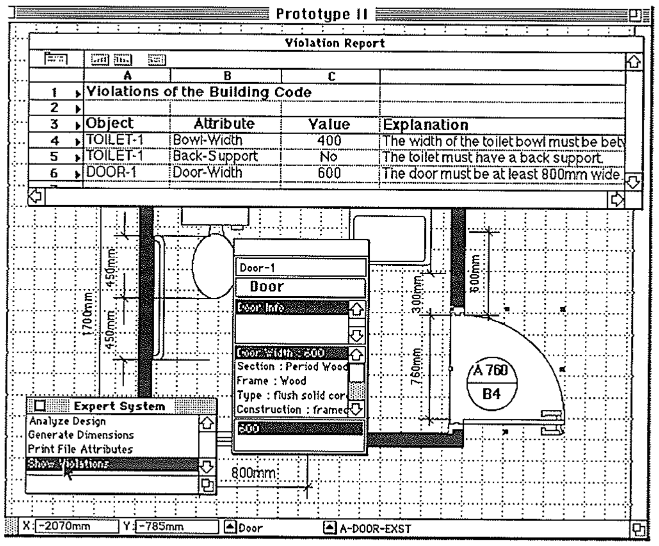


FIGURE VII.20. - Command Palette

record in the background. All of the related door type data is retrieved from and stored in a separate database(s). The retrieved data, not the coding, is displayed to the user on the screen.

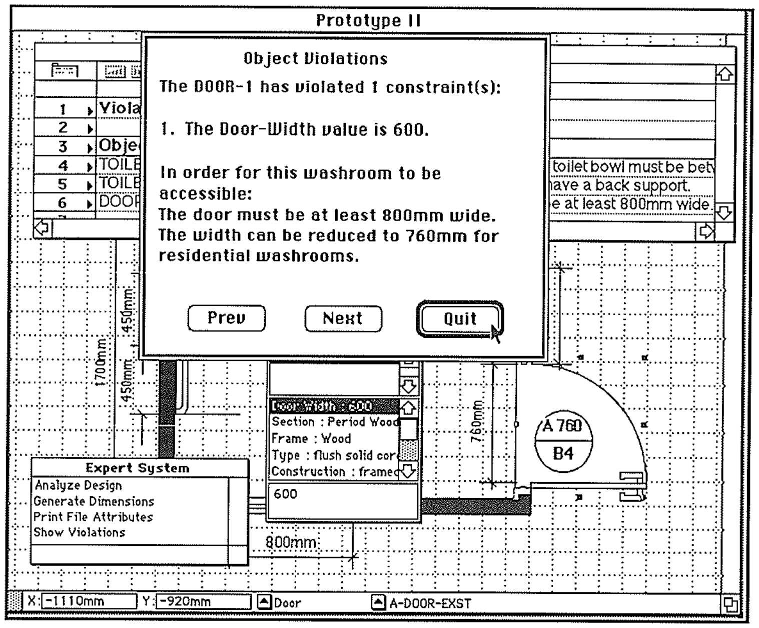
VII.F.7. Providing user feedback. The prototype presents several types of comments to the user. A Violation Report is returned to the CAD tool and automatically appears on the top of the screen where it can be repositioned, closed, or reopened as many times as desired. This report contains specific violations based on the building code for the objects in the design, including a brief explanation of an acceptable value for the object. Each object that has one or more violations associated with it is highlighted in the drawing (Figure VII.21.).



	A	B	C	
1	Violations of the Building Code			
2				
3	Object	Attribute	Value	Explanation
4	TOILET-1	Bowl-Width	400	The width of the toilet bowl must be bet
5	TOILET-1	Back-Support	No	The toilet must have a back support.
6	DOOR-1	Door-Width	600	The door must be at least 800mm wide.

FIGURE VII.21 - Violations report worksheet

An additional command, Show Violations, can be used to provide the user with more extensive information about a specific object violation (Figure VI.22.). The designer is not forced to change the design. However, the Analyze Design command can be re-invoked if design changes are made and further analysis is desired (Hill, et al. 1993).



Object Violations

The DOOR-1 has violated 1 constraint(s):

1. The Door-Width value is 600.

In order for this washroom to be accessible:
The door must be at least 800mm wide.
The width can be reduced to 760mm for residential washrooms.

1	Viola
2	
3	Obje
4	TOILE
5	TOILE
6	DOOR

toilet bowl must be bet
ave a back support.
e at least 800mm wide.

Door Width : 600
Section : Period Wood
Frame : Wood
Type : flush solid cor
Construction : framed

600

FIGURE VII.22. - Show Violations

VII.G. Summary and Observations

The prototype Smart Graphic Environment developed in this study is by no means a complete system. Prototyping is a common technique used by KBS developers to help identify where key problems lay. Indeed, the prototypes created in this research provided substantial insight into the potential of using an object-based approach to integrate knowledge base evaluation within a CAD design environment.

Through the process of conceptual modeling of building code constraints the strong hierarchical nature of architectural knowledge becomes evident. Building the initial knowledge base using an expert system shell allowed Prototype I to be implemented relatively quickly. This was an important factor in enabling the programmers to receive feedback from the domain experts early in the implementation process.

The experience of developing Prototype I revealed that although it is possible to encode building code constraints relating to special washrooms using a rule-based approach, an object-based approach is far more suitable for the representation of architectural knowledge. The exchange of data between the knowledge base and the CAD environment for the purpose of evaluation depends on a shared definition of building design objects and concepts. An object-based approach creates a model of knowledge related to building design objects. Whereas the requirements for objects are subject to change over time, the objects themselves (wall, door, roof) are less likely to change. In this manner, object-based approach is able to establish a more comprehensive and stable framework for representing architectural knowledge and data found within both the constraint and CAD building models.

The objective of Prototype II was to create a smooth exchange of data between the knowledge base, integrated graphic database and the designer using an object-based approach. Rather than trying to adjust to the pre-established inferencing and reasoning strategies imbedded in the expert system shell, custom building the knowledge base and control structure in LISP offered considerably more flexibility. The implementation of Prototype II led to a better understanding of the development of CAD models for the purposes of knowledge base evaluation as well as providing feedback based on that evaluation in a manner that is useful to the designer.

There are three areas of exploration in terms of developing the prototype further:

(1) user interface,

(2) expanding the knowledge base, and

(3) adding new knowledge.

<u>VII.G.1. Improving user interface.</u> User interface is terminology used to describe features of a computer system that facilitates human-machine interaction. In the prototype Smart Graphic Environment such features were needed to facilitate the exchange of information between the architect user and the electronic expert that specializes in the building code evaluation of heritage buildings. In order to make the system easy to use, a conscious effort was made to create an interface that closely paralleled procedures that architects already would be familiar with. Like a human consultant asking for drawings, the KBS requires that the architect create a building design record. However, instead of using pen and paper, drawings are created digitally. Instead of calling the consultant on the telephone, the designer initiates an evaluation by clicking on the appropriate command on the computer screen without leaving the drawing environment. Feedback is structured to identify each violation in the drawing by selecting the problem object (similar to red-lining), and provide a report stating the existing condition and explaining the building code requirement.

There are several areas for improvement in the current user interface and control. A printout of the violation report, including a list of all of the relevant articles of the building code identified, should be provided as a reference for the architect. Currently the system only evaluates the entire design. User control of the analysis should be improved to provide analysis by (1) object, (2) entire design, or (3) evaluation type (based on building code, historical building regulation and/or user requirements). The user should have the option to query the system about desirable object characteristics before they begin to design (Figure VII.23.).

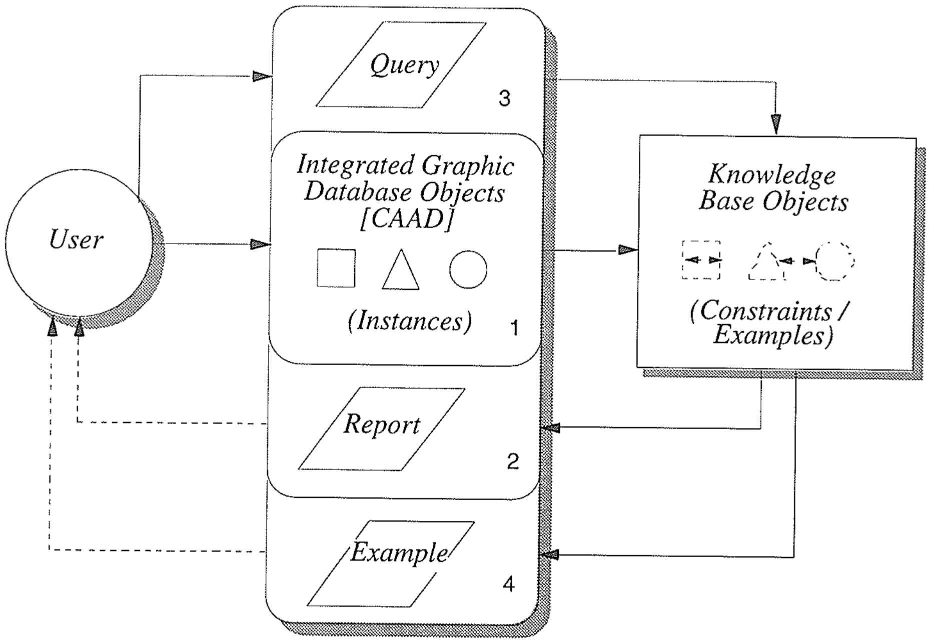


FIGURE VII.23.- Facilities for improving prototype user interface

The system was designed to be used by an architect, however, it does not discriminate between the novice and expert user. This issue is significant in terms of assisting the user in creating the CAD heritage building record and AAC. A person who is not familiar with the building code may not understand the terminology or be able to appropriately assess attribute values (such as performance). The addition of an on-line "help" facility as well as written documentation in the form of a user manual are two potential methods for providing assistance.

The current prototype relies on the user to create a small CAD file and database containing the objects and object descriptions required for the analysis. This task is relatively straightforward given the simplicity of the initial design model -- a washroom. MiniCAD+^{TM} provides a variety of drawing and editing tools as well as a data palette for inputting information directly into pre-defined fields of the IGDB. The prototype digital heritage record contains only two-dimensional drawings, plans and elevations. The next stage of development should explore the use of three dimensional CAD objects. This type of representation would provide many more opportunities for creating virtual models appropriate for visualization as well as evaluation.

It is difficult to assess at what point that the task of creating a large digital heritage building record would become so time-consuming that it outweighed the benefits of knowledge base evaluation. The creation of a CAD file and IGDB for an entire building would obviously be more difficult than a single room. In terms of generating a geometric model, semi-automatic generation of CAD objects using certain digital recording techniques, such as digital photogrammetry, could be used (as discussed in Chapter V). The use of predefined layers, database records and fields can be effective means of assisting the designer in the development of a semantic description. The use of a relational database would mean that this information would be stored, accessed, and retrieved efficiently.

The current lack of an industry standard for building product data models for classifying and organizing building data is an impediment to the exchangeability of models between different computer systems. However, one of the discoveries of this research is that the input data requirements came naturally out of the KBS development. The building code is designed to be able to encompass a variety of building designs. Therefore, building code conceptual constraint models have the potential ability to classify a lot of different configurations and instances of objects under a few classifications. By attempting to achieve exchangeability between the CAD and KBS for building code evaluation a common definition for building product models becomes clearer. Future research in this area will inevitably identify building data for pre-defined data fields and records.

In the prototype the designer had no difficulty in properly creating and attribute coding all of the objects for a digital heritage washroom design. The use of *certainty* *factors*, considered unnecessary for the prototype room model, may be necessary in a KBS that performs a more comprehensive building code analysis. Certainty factors are used where data may be incomplete or when the user is unsure of the accuracy of the data. This is particularly important if the system relies on the user's experience to judge actual or equivalent performance. Take for example a fire code analysis. An estimation of how long in terms of hours an existing wall will provide fire separation can be reasonably determined on careful consideration of external evidence, calculations, and the performance of similar walls. Such an assessment may be able to predict with 90% certainty the actual performance of the wall in a fire situation. As it is impractical to burn down an existing building to see if it meets fire code requirements, a certain tolerance level of uncertainty is allowed, given the reasonable accuracy of other analysis techniques. Certainty factors are a means of statistically indicating how conclusions made by the KBS may not be absolutely accurate.

On-site testing by a representative group of users would be helpful to gain further insight into how well the concepts presented here have been implemented. Findings could then be incorporated those into future versions.

VII.G.2. Expanding the knowledge base. The task of selecting and applying the NBCC requirements for the accessibility of washrooms lends itself well to expert system techniques. In general the requirements are fairly explicit and, like other types of building code analysis, can be applied in a logical and sequential fashion. Based on the knowledge contained in the NBCC it is possible to create conceptual models of the constraints, and in so doing establish a minimum ideal or *As-Required* building constraint model. This model can then be compared against an existing or *As-Designed* building model. The data requirements for the knowledge base are easily established based on conceptual representations. Understanding the data requirements of the KBS is a necessary step in determining what information could be supplied by a CAD model and database.

Although there were only seven key objects (toilet, sink, door...etc.) involved in the analysis of the special washroom, the configurations of these objects can vary considerably from building to building. The ability to consider a number of arrangements with only a few constraints makes the system interesting without becoming unmanageable.

The prototype Smart Graphic Environment (Prototype II) can complete an accessibility analysis of a representative washroom plan (Figure VII.24.) in five minutes on the Macintosh IICi with three MGs of RAM. The prototype knowledge base incorporates approximately approximately sixty constraints out of close to 3000 constraints that are found in the entire NBCC.† Although the selected constraints are representative of typical NBCC requirements, it is difficult to determine with accuracy how the system would deal with both a larger rule-base and larger IGDB.

The key impediment to the extensibility of the system is knowledge engineering. Conceptually modeling and representing within the knowledge base the entire NBCC is a huge task. Although the requirements of the NBCC are well-documented and its information is relatively stable, periodic changes to the Code would require adjustments to the knowledge base. A sophisticated control strategy would also have to be integrated to manage and maintain the larger knowledge base.

† The complete KBS LISP program code for Prototype II is located in Appendix D.

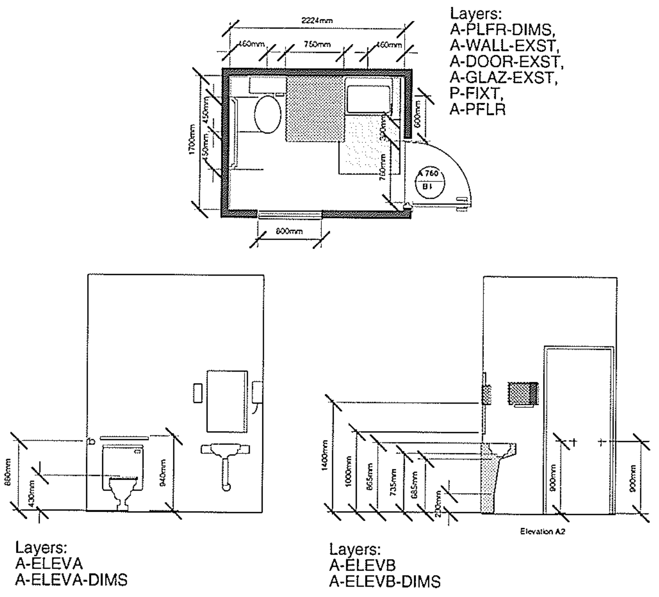


FIGURE VII.24. - Prototype II test washroom design

VII.G.3. Adding new knowledge. The prototype follows a *verification* paradigm. The expertise in the knowledge base can be used to identify the specific requirements applicable to the building or object in question from a number of possibilities as represented by the entire building code. The domain knowledge helps eliminate inapplicable requirements and uses applicable requirements that do apply to diagnose violations. Depending on the violation, further explanation or examples can be used to assist the designer in the resolution of the violations. This approach is fairly representative of a human code expert analysis protocol.

Based on the initial experience with the prototype system it was felt that, in the case heritage buildings, it would be expedient to integrate a means of building or object classification to determine historical value before a building code evaluation (currently the system relies on the user to provide this information). Building code violations involving objects with a high historical value require special consideration. Therefore, future prototype development should include the addition of new knowledge to the system from a different domain - that of an architectural historian.

Knowledge base systems that contain the expertise of more than one domain are considered to be *multi-agent*. This means the system is capable of thinking about a problem from different perspectives. Therefore it must also have the facility for the handling conflicting goals between agents. By understanding an architectural object has both historical value and code implications the system can then begin to consider strategies for resolution of problems. One technique may be to encode methods for dealing with common code violations found in heritage buildings within the knowledge base. When relevant this information can be presented to the user as part of the violation report.

<u>VII.H. References</u>

Ballantyne, D. 1983a. "Priority problems and typical solutions." *Accommodation of Disabled Visitors at Historic Sites in the National Park System.* Part 1, Chapter 6. Washington, D.C.: National Park Service. 15 - 22.

Ballantyne, D. 1983b. "Typical solutions to typical problems." *Accommodation of Disabled Visitors at Historic Sites in the National Park System.* Part 2, Chapter 5. Washington, D.C.: National Park Service. 33 - 47.

Björk, B. and K. Kähkönen. 1991. "Preface." *VTT Symposium - Computers and Building Regulations.* Espoo, Finland: Technical Research Centre of Finland. 27 - 29 May. 3-6.

Carrico, M., J. Girard, J. Jones. 1989. *Building Knowledge Systems.* New York: McGraw-Hill.

Canadian Commission on Building and Fire Codes [CCBFC]. 1993. *Guidelines for Application of Part 3 of the National Building Code of Canada to Existing Buildings.* Ottawa: National Research Council of Canada.

Canadian Commission on Building and Fire Codes [CCBFC]. 1990. *National Building Code of Canada.* Ottawa: National Research Council of Canada.

Canadian Paraplegic Association, Manitoba Division [CPA]. 1989. *ACCESS - a Guide for Architects and Designers.* Winnipeg, MB: CPA.

Canadian Standards Association [CSA]. 1990. *CAN/CSA-B651-M90 Barrier-free Design - A National Standard for Canada.* Ottawa, ON.: CSA.

Cornick S, D Leishman, R Thomas. 1991. "Integrating building codes into design systems." *VTT Symposium - Computers and Building Regulations.* K. Kähkönen and B. Björk (eds.). Espoo, Finland: Technical Research Centre of Finland. 210 - 236.

de Gelder J. and L. Lucardie. 1993. Knowledge and data modeling in CAD/CAM applications. *Design and Decision Support Systems in Architecture.* H. Timmermans (ed.). The Netherlands: Kluwer Academic Publishers. 111 - 122.

de Gelder J and L. Lucardie. 1994. "What conceptual modeling is and isn't." presented at *Design and Decision Support Systems 1994.* 14 - 19 August , 1994. Vaals, The Netherlands.

de Waard M. and F. Tolman. 1991. "Modeling of building regulations." *VTT Symposium - Computers and Building Regulations.* K. Kähkönen and B. Björk (eds.). Espoo, Finland: Technical Research Centre of Finland. 195 - 209.

Di battista G., H. Kangasallo, R. Tammasia. 1989. "Definition libraries for conceptual modeling." *Data & Knowledge Engineering.* 4. 245 - 260.

Dym, C., R. Henchey , E. Delis and S. Gonick. 1988. "A knowledge-based system for automated architectural code checking." 20:3 (April) Butterworth & Co. Ltd. 137 - 145.

Frye, J. 07 November 1991. Personal interview.

Frye, M., D. Olynick, R. Pinkney. 1992. "The development of an expert system for the fire protection requirements of the National Building Code of Canada." (Joint CIB Workshops) *Integrated Computer Aided Design and Computers and Building Standards.* 12 - 14 May. Montreal.

Hattis, D. 1981a. "How existing buildings and building rehabilitation are regulated." *Association for Preservation Technology Bulletin.* XIII:2. Champaign, IL: APT. 8 - 12.

Hattis, D. 1981b. "Some current regulatory innovations related to building rehabilitation." *Association for Preservation Technology Bulletin.* XIII:2. Champaign, IL: APT. 13 - 16.

Hansen, A.T. 1984. "Applying building codes to existing buildings." *Canadian Building Digest.* (January) Ottawa: National Research Council of Canada. CBD 230 - 230 - 4.

Hill, S., M. Evans, L. Strachan. 1992. "Overview of an object-based approach for automated expert evaluation of historical structures." Presented at *Computers in Conservation.* 16 - 18 August 1992. Québec City: ICOMOS [forthcoming in APT *Bulletin*, the Journal of the Association for Preservation Technology].

Hill, S., M. Evans, L. Strachan, P.R. Perron. 1993. "The use of smart graphics for the redevelopment of historical structures." presented at *Design Decision Support Systems in Architecture.* 6 - 10 July 1992. Eindhoven, The Netherlands [pending publication in the Journal of Architectural Planning and Research].

Hosking, J., W. Mugridge, J. Hamer. 1991. "An architecture for code of practice conformance systems." *VTT Symposium - Computers and Building Regulations.* K. Kähkönen and B. Björk (eds.). Espoo, Finland: Technical Research Centre of Finland. 171 - 180.

Institute for Research in Construction, National Research Council of Canada [IRC]. 1993. *Construction Resources (Beta 1.0 Version).* (CD-ROM) Ottawa: Megalith Technologies Inc.

Jester, T. and J. Hayward (eds). 1993. *Accessiblity and Historic Preservation Resource Guide.* (Conference Proceedings) 24 - 25 June. Chicago.

Lucardie, G.L. 1993. "A functional approach to realizing decision support systems in technical regulation management for design and construction." *Design and Decision Support Systems in Architecture.* H. Timmermans (ed.). The Netherlands: Kluwer Academic Publishers. 97 - 110.

Markman, H. 1981. "Fire ratings of archaic materials and assemblies." *Association for Preservation Technology Bulletin.* XIII:2. Champaign, IL: APT. 19 - 22.

Mitusch, P. 1991. "Expert system for the Norwegian building regulations - a new approach." *VTT Symposium - Computers and Building Regulations.* K. Kähkönen and B. Björk (eds.). Espoo, Finland: Technical Research Centre of Finland. 36 - 42.

Nijessen, G.M. and T.A. Halpin. 1989. *.Conceptual Schema and Relational Database Design, a fact oriented approach.* Sydney, Australia: Prentice Hall.

Parrott, C. 1980. *Access to Historic Buildings for the Disabled.* Washington, D.C.: National Park Service, U.S. Dept. of Interior.

Prudon, T. 1990. "Preservation vs. codes." *Architectural Record.* 179 No. 3, NY: McGraw Hill Inc. 52 - 54.

Prudon, T. and S. Dalton. 1981. "The accessible path in historic properties." *Association for Preservation Technology Bulletin.* XIII:2. Champaign, IL: APT. 31 - 39.

Schley, M.K. 1990. *CAD Layer Guidelines - Recommended Designations for Architecture, Engineering, And Facility Management Computer Aided Design.* New York: American Institute of Architects Press.

Smith, W. and T. Frier. 1990. *Access to History: A Guide to Providing Access to Historic Buildings for People with Disabilities.* Boston: Massachusetts Historical Commission.

Sowa, J.F. 1984. *Conceptual Structures: Information Processing in Mind and Machine.* Don Mills, Ontario: Addison-Wesley.

Taremäe, A. 1991. "Expert system for structural analysis." *VTT Symposium - Computers and Building Regulations.* K. Kähkönen and B. Björk (eds.). Espoo, Finland: Technical Research Centre of Finland. 43 - 44.

Tuominen, P. 1991. "Fire Expert - an expert system for fire safety regulations in building design." *VTT Symposium - Computers and Building Regulations.* K. Kähkönen and B. Björk (eds.). Espoo, Finland: Technical Research Centre of Finland. 48 - 52.

Turk, Z. 1991. "Building model standard as a foundation for computer integrated design." *VTT Symposium - Computers and Building Regulations.* K. Kähkönen and B. Björk (eds.). Espoo, Finland: Technical Research Centre of Finland. 181 - 194.

Turner, J.A. 1991. "Guide to reading NIAM diagrams." (paper) University of Michigan.

Turner, J.A, J. Wix, C. Prayhoon. 1991. "Spatial systems model." (paper) University of Michigan.

Vanier, D. 1991a. "A parsimonious classification system to extract project-specific building codes." *VTT Symposium - Computers and Building Regulations.* K. Kähkönen and B. Björk (eds.). Espoo, Finland: Technical Research Centre of Finland. 134 - 145.

Vanier, D. 1991b. "Bibliography of research in automated building codes." *VTT Symposium - Computers and Building Regulations.* K. Kähkönen and B. Björk (eds.). Espoo, Finland: Technical Research Centre of Finland. 250 - 259.

CHAPTER VIII

CONCLUSION

In this era of finite resources and shrinking economies it would be unwise for practicing architects not to recognize the growing trend away from new construction towards rehabilitation of existing buildings. In the face of this change is a need to develop strategies for efficiently documenting, analyzing, and adapting existing architectural resources. A particular challenge is the redevelopment of heritage buildings, where integration of a new use must be done without destroying important historical features, while at the same time accommodating modern conveniences and performance requirements. This requires a number of specialized skills and domain expertise that are not commonly found in most architectural practices. Now that sophisticated computer technology is affordable and readily available it is only natural to research how we can better utilize computers in the field of conservation.

Most of the recent research on computers in conservation is in relation to recording technologies. It is important to recognize that as more and more records are put into digitized formats that this information can be reused for a variety of purposes. Heritage recording is one of two primary conservation activities, the other being analysis. The expert analysis of existing building conditions provides the understanding required to make appropriate redevelopment design decisions. Therefore, design decision-making requires (1) the management of building information and the (2) utilization of expert knowledge.

This study explores the development of a special type of Intelligent Computer Aided Design system called a Smart Graphic Environment. In this environment computer technologies provide a means of not only storing heritage building information, but expert knowledge as well. The integration of these technologies provides design decision-making support for the redevelopment of heritage structures.

An effective design decision-making environment relies not only on individual skills, but the ability of its team members to be able to interact and share information. One of the primary challenges of this research was to determine ways to facilitate communication between the components of the Smart Graphic Environment. Each component of the environment uses its own language for design; CAD systems generally use geometry, KBS use rules and constraints, and designers use concepts. However, by adopting an object-based approach, where domain knowledge is linked with data objects, a shared "language" of design discourse is created. Knowledge is attached to objects in both the drawing (in the form of attributes) and the knowledge base (in the form of rules and constraints). A knowledge level approach allows the representation of design according to the conceptual models used by designers and other domain experts.

One of the important characteristics of the Smart Graphic Environment is that the Designer is provided access to expert knowledge through the CAD environment. Designers are used to communicating their ideas through drawings. However, much of the information, with the exception of geometric information, contained within CAD drawings is implicit and therefore has to be interpreted in some manner in order for the computer to evaluate it. This meant overlaying object-based descriptions and attaching attributes to raster and vector-based graphics. In this way, the facts about the design can be automatically extracted from the CAD model and IGDB to be analyzed by the knowledge base system.

One of the shortcomings of the Smart Graphic Environment is that it relies heavily on the ability of the designer to interpret the design information for the system. The reason for this is due, in part, to the current lack of adequate industry standards both for the production of digital graphics and product data models. One of the interesting discoveries of this study was that out of the development of the knowledge base object constraints, key elements were identified for any given evaluation. As a result, depending on the different level or type of evaluation one can priorize when and what information to include in the digital record of the building.

The Smart Graphic Environment described in this study does not provide for the automatic generation of redevelopment solutions. To a large extent this is simply not feasible given today's technology. Although KBS are capable of handling and maintaining over time large amounts of information, they work best when given very specific problems to solve within specialized areas of expertise. Used interactively, KBS are able to support the designer by imparting a deeper understanding of key issues and constraints relating to a particular design problem so that the architect can apply their own inherent skills and experience to come up with creative or innovative design solutions.

The development of prototypes is, by far, the most effective means of evaluating the effectiveness and practicality of design decision support systems. In this study prototypes provided valuable insight into key problems as well as potential opportunities for the Smart Graphic Environment. Prototype I, a system developed using a rule-based expert system shell, revealed the potential value of the system. Prototype II, developed using object-based approach and integrated within a CAD system, demonstrated considerably more flexibility and an improved user environment. Important aspects for future study are: user interface, expansion of the knowledge base, and the addition of new knowledge types.

This research represents only one of a growing number of contributions to the study of intelligent design systems. One of the problems with the creation of such a system is the high expectations of what computer systems can do. Computers are not a panacea, but merely one of many tools, albeit a powerful one, at the experienced designer's disposal. One does not expect a hammer to rise from the tool box and build a house by itself, nor should a designer expect that a computer will solve all design problems. From the experience of this research, computer technologies demonstrated tremendous capabilities for acquiring and storing large amounts of heritage building information. They also can be relied upon to unerringly and tirelessly apply encoded knowledge to performing analysis tasks that they are directed to do. However, it was the human experts that were able to deal with the unusual questions and provide the most creative and valuable insights. In the end, "smart" systems enable experienced designers to do what they already do, only better.

APPENDICES

APPENDIX A

TRIGGERING CODE REQUIREMENTS FOR APPLICATION TO EXISTING BUILDINGS

A.1 Identifying Code Requirements

"Triggering" is a term that refers to an event that mandates the application of new construction code requirements onto a rehabilitation project. A KBS can use these rules to determine if the building is excluded from having to comply with the requirements of the building code. There are four triggering mechanisms for the NBCC:

(a) Renovation or Alteration

(b) Change in Major Occupancy

(c) Change in Owner

(d) Retroactive or Maintenance Law

(e) Guidelines

A.1.a. Renovation or Alteration. When an owner receives a building permit for an addition to a building either horizontally or vertically they will be required to upgrade the entire building to current construction code requirements. If the proposed addition is small the local authority may require that only the addition be compliant. Some horizontal additions are also exempted when they are separated from the existing building by a firewall. Interior alterations are required to meet new code requirements. However, this does not mean that the entire building has to be upgraded if, for example, a single floor is affected. The alteration must not contribute to further non-compliance of other areas (i.e. block off egress routes).

Some of the model building codes used in the United States utilize what is known as the 25 - 50% rule. This rule states that when a rehabilitation project exceeds 50% of the building property value the entire building must be upgraded to modern building code requirements. "The original purpose of the rule was to prevent rather than promote the rehabilitation of certain classes of buildings" (Hattis 1981a, 11). States which have adopted local guidelines for applying building codes to heritage properties tend to dispute that these numbers are arbitrary and that they discriminate against lower valued or small buildings, and therefore have adopted more flexible approaches (Ibid.).

A.1.b. Change in Major Occupancy. When the major occupancy of a building is proposed, such as when a warehouse (F-2) is being changed into a restaurant (A-2), the entire building must meet new construction standards. Where there is a multi-major occupancy, each area must be made to comply as if the entire building was of that occupancy (Figure A.1.). Usually, it is advisable that the building be upgraded to the most stringent code requirement to allow for the most flexibility. For example, if the building has a restaurant on one floor and an office area on the second, the owner will have the option of leasing each floor to either occupancy if the building meets the code requirements of both. This may not always be cost effective in heritage buildings.

From time to time an owner will be allowed to upgrade only that portion of their property which is occupied. One of the particular problems of allowing a piecemeal upgrading is that it is a short-term solution. It may appear reasonable that a tenant only using a small portion of a very large warehouse should only have to upgrade the area they are in. The problem is when this is allowed again for another tenant and so on. The lack of a cohesive planning approach means potential conflicts of egress routes, access to washrooms and fire fighting equipment and so on -- just because an adjacent tenant locks a door or poorly places a partition. An owner who adopts a more holistic planning approach to their building is more likely to save money in the long term. A building which is retrofitted in its entirety benefits from the input of architects, users, owners, building managers, plan examiners, etc., at the onset, who can deal with the large issues of fire separations and accessibility. This leads to the development of a maintainable, well-planned solution which can, in turn, attract new tenants.

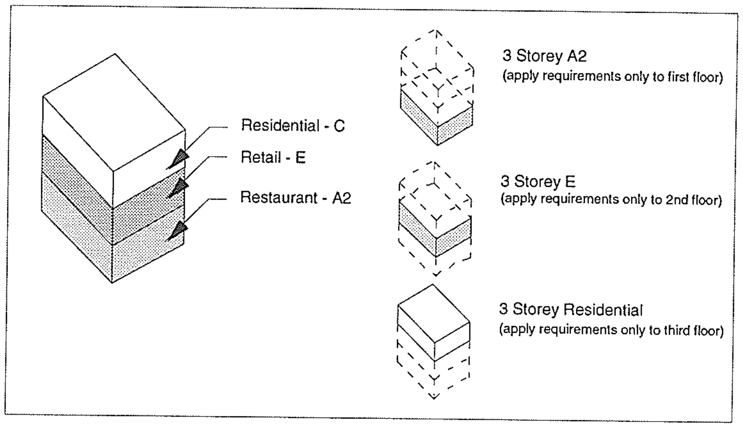

Residential - C
Retail - E
Restaurant - A2
3 Storey A2
(apply requirements only to first floor)
3 Storey E
(apply requirements only to 2nd floor)
3 Storey Residential
(apply requirements only to third floor)

Figure A.1. - Multi-occupancies

<u>A.1.c. Change in owner.</u> Normally when a building changes owner but not use then the building falls under a "grandfathering" of code requirements (non-conformity allowed). It has to be maintained according to the level of the building codes to which it was constructed or upgraded to previously. This rule will not apply if the owner intends on increasing occupant load or some such similar change which will affect the performance criteria.

A.1.d. Retroactive or maintenance law. Some laws were created to insure that buildings are operated and maintained properly. Municipal governments will sometimes enact by-laws which will mandate acceptable levels of safety for its citizens.

The City of Winnipeg Residential Upgrading By-Law 4304 applies to any residential occupancy larger than a single family dwelling. It particularly impacts on apartment blocks and converted single family dwellings into multi-tenant or small businesses or a combination thereof. It is a retroactive law that requires all such buildings be upgraded to an acceptable level. The by-law provides for a phased upgrade in the form of allotted times for compliance with each requirement. The bylaw specifications range from putting in adequate fire alarms to substantial renovation of means of egress equivalent to today's standards.

Maintenance and Occupancy municipal bylaws deal only with the protection of the public and property. A building can be left abandoned as long it does not present a hazard of collapse, is secured properly, and the exterior kept tidy. These by-laws do not require an owner to maintain or even heat the interior. This particular omission became a matter of some debate in 1990, when the City of Winnipeg closed its central steam plant which served approximately 30 heritage properties in the downtown area. Since that time only 2 - 3 of these buildings have been provided with an alternate heating system. It is anticipated that without a providing encouragement in the form of tax incentives or a bylaw mandating the maintenance of heating these properties, left vacant and exposed to harsh winter conditions will quickly deteriorate.

The fire department periodically inspects buildings (as do other department such as Health and Labour). Although they have the authority to do so, fire inspectors will rarely exercise their right to enforce stringent code compliance to the *National Fire Code of Canada*. Often compliance is based on the willingness of the owners to act voluntarily. When the fire chief discovers an area of concern he/she can ask city hall to pass municipal by-laws, such as a one on standpipes, could demand owners to immediately upgrade of unacceptable systems or face court proceedings. Retroactively applied laws are very unpopular because of the financial burden placed on owners of older buildings. By-laws usually will provide for phased upgrading, thereby offsetting costs over a longer period of time.

When a fire occurs in Winnipeg, MB, for example, the Winnipeg Building Branch and the Fire department have classification system which they apply. The five level fire classification is based on an on-site inspection and its mandate ranges from minor repair to demolition. In the former case a building only needs to be secured and returned to the way it was, in the latter case very extreme damage would warrant the demolishing of the building. If any substantial repairs are warranted architectural drawings/specifications will have to be prepared to satisfy the fire department that the work is done in an acceptable manner.

A.1.e. Codes and Guidelines. The NBCC Section 2.5 "Equivalents" allows for the adoption of alternative materials or methods of design other than those described in the Code based on a supplied evidence of equivalent performance (such as a legitimate code used in another province). In response to the growing renovation/rehabilitation industry both Manitoba and the NRC plan to integrate methods for adopting the building code to existing buildings within their respective building codes .

An owner who is having difficulty with compliance of a building will be to referred to other building codes or guidelines. One Canadian example is British Columbia Building Code's Subsection 3.7.4. "Alterations and Additions to Existing Buildings" on alternative requirements for upgrading of existing buildings.

Most states in the U.S. have adopted one of three model codes including one produced by the Building Owners & Code Administers [BOCA]. In 1984, BOCA created the *National Existing Structures Code* as part of their Basic/National Codes.

Some municipalities also have created special codes and created review boards for dealing with rehabilitation, particularly with residential occupancies. These special provisions described in (Hattis 1981b) and includes: Washington, San Francisco (hazard matrix), Denver (small building exemption), Massachusetts (Article 22 - occupancy is classified according to hazardous conditions), and California (created a separate building code for heritage properties).

The U.S. Department of Housing and Urban Developed [HUD] published *Rehabilitation Guidelines* in 1980 to help "modify existing building code and building regulatory process to facilitate rehabilitation in their cities; and to assist rehabbers, architects and engineers who have specific rehabilitation/building code problems" (Kapsch, 1981b). The guidelines are not codes, but are intended to be used to help with the application of modern building codes in existing buildings through clarification of intent and examples.

Guidelines focus on issues such as the cost-benefit factor -- when the costs of upgrading severely outweigh the achieved benefits. In heritage buildings one of the "costs" may be the loss of an essential historic feature which is not easily quantifiable. The authority having jurisdiction has the discretion of weighing these factors. Sometimes they will tell the owner get an opinion of a specialist or involve the persons with disabilities and/or the heritage community in resolving design issues.

A.2 References

Frye, J. 28 April 1992. Personal Interview.

Hattis, D. 1981a. "How existing buildings and building rehabilitation are regulated." *Association for Preservation Technology Bulletin.* XIII (2). 8 - 12.

Hattis, D. 1981b. "Some current regulatory innovations related to building rehabilitation." *Association for Preservation Technology Bulletin.* XIII (2). 13 - 16.

Kapsch, R. 1981. "Cracking the codes." *Association for Preservation Technology Bulletin.* XIII (2). 5 - 7.

APPENDIX B

PROTOCOLS

The protocol developed for the prototypes in this study are a hybridization of two existing prototypes by (Markman 1981) and (Dym, et al. 1988). In the former case, the protocol was developed as an approach to dealing with the application of modern building code performance standards to older building materials and construction types. In the latter case, a protocol was generated in conjunction with the development of a KBS that deals with the application of building code requirements for the preservation of life safety.

The protocol found in (Markman 1981) is as follows:

> "The process of evaluating existing construction consists of four distinct phases, which are described below:
>
> - Identify code requirements
> - Identify existing materials and construction
> - Estimate level of performance
> - Develop design solutions" (19)

The protocol for code compliance evaluation developed by (Dym, et al. 1988) for application of the Life Safety Code is more detailed and comprehensive:

> "The review process has three basic procedures:
>
> - identification of the parts of the building

- selection of applicable provisions of the LSC
- application of those provisions to a particular situation" (138).

High level protocol for code checking process:

Step 1	Establish number of stories, siting, overall shape
Step 2	Establish occupancies
Step 3	Establish and check fire separation zones
Step 4	Establish types of construction
Step 5	Establish egress routes and check width
Step 6	Establish sprinkler presence or absence
Step 7	Establish and check hazardous areas
Step 8	Check enclosure of corridors
Step 9	Check type and number of exits
Step 10	Check travel distances
Step 11	Check for egress to grade
Step 12	Check widths of all doors
Step 13	Establish and check smoke zones
Step 14	Check finishes

(Ibid.)

REFERENCES

Markman, H. 1981. "Fire ratings of archaic materials and assemblies." *Association for Preservation Technology Bulletin.* XIII (2) 19 - 22.

Dym, C.L., R.P. Henchey, E.A. Delis, and S. Gonick. 1988. "A knowledge-based system for automated architectural code checking." *Computer Aided Design.* 20 (3) April. Butterworth & Co. Ltd. 137 - 145.

APPENDIX C

FRAMES

The following are frames containing special washroom object constraints based on the NBCC. These were created as part of the implementation of the prototype systems.

BUILDING				
SLOT	VALUE(S)	POSSIBLE VALUES	LEGAL VALUE(S)	VALUE TYPE
major occupancy	mercantile	residential, low and medium hazard industrial, high hazard industrial, assembly, institutional, business and personal services, mercantile, other.	ONE OF (?V = business and personal services, institutional, low and medium hazard industrial, assembly, mercantile)	list
area of floor	600	n/a	?V=>600	numeric (m2)
nature of project	change occupancy	renovation of existing, change of occupancy, no change, new construction	ONEOF (?V = renovation of existing, change of occupancy)	list

WASHROOM				
SLOT	VALUE(S)	POSSIBLE VALUES	LEGAL VALUE(S)	VALUE TYPE
Use Space	public special washroom	public special washroom, public washroom, private washroom, other	?V = public special washroom	text
Use Space Length (sp w.c.)	1700	n/a	?V => 1700	numeric (mm)
Use Space Width (sp w.c.)	1700 min	n/a	?V => 1700	numeric (mm)
Circulation	barrier-free path of travel	barrier-free path of travel, other	?V = barrier-free path of travel	text

SPECIAL WASHROOM DOOR				
SLOT	VALUE(S)	POSSIBLE-VALUES	LEGAL VALUES	VALUE TYPE
door clear width	800	n/a	(?V>=800) OR (?V>= 760) AND Residential	numeric (mm)
space pull	600	n/a	?V >= 600	numeric (mm)
space push	300	n/a	?V >= 300	numeric (mm)
swing	into exit	into exit, into room	?V = into exit	text
open force	38	n/a	?V <= 38	numeric (N)
closing period	3	n/a	?V >= 3	numeric (sec)
ht. threshold	13	n/a	?V <= 13	numeric (mm)
knob type	lever	lever, pushplate, panic bar, round knob, canted, "D" pull	ONEOF(?V, (lever, push plate, panic bar, "D" pull)	list
lock type	slider	graspable latch, flip, slider, dial	ONEOF (?V, (graspable latch, flip, slider))	list
lock ht. lctn	900	n/a	?V>= 900 AND <=1000	numeric (mm)
door pull	yes	yes, no	(?V=Yes) AND (doorswing=out) OR (?V=No) AND (doorswing =in)	text
door pull length	140	n/a	?V>= 140	numeric (mm)
door pull ht. lctn	900	n/a	?V>= 900 AND <=1000	numeric (mm)
coathook	Yes	Yes, No	?V = Yes	text
coathook ht. lctn	1400	n/a	?V<=1400	numeric (mm)
coathook dept	25	n/a	?V<= 25	numeric (mm)
signage	Yes (International Symbol of Accessibility)	Yes, No	?V = Yes	text

WATERCLOSET				
SLOT	VALUE(S)	POSSIBLE-VALUES	LEGAL VALUES	VALUE TYPE
seat height	430	n/a	?V>=400 AND <=460	numeric (mm)
bowl width	490	n/a	?V>=525 AND <= 460	numeric (mm)
back support	Yes	Yes, No	?V = Yes	text
spring seat	No	Yes, No	?V = No	text
water tank	Yes	Yes, No	(?V = Yes) OR (?V = No AND rear grab bar)	text
mount	Wall	Wall, Floor	?V = Wall) OR ?V = Floor AND =300 use space depth**	text
control lctn	openside tank	openside tank, openside wall, close die wall, close tank, floor	ONEOF(?V, openside tank, openside wall)	list

control type	lever	push button, lever, automatic, chainpull, footpedal, other	ONEOF (?V, pushbutton, lever, automatic)	list
conrol force	5	n/a	?V<= 5	numerric (lbf)
width lctn	475	n/a	?V>=460 AND <= 480	numeric (mm)
clearspace width	750	n/a	?V >= 740 AND <= 760	numeric (mm)
clearspace width	840	n/a	?V >= 675 AND <= 840	numeric (mm)
toilet paper disp. type	continous feed	??	?V = continuous feed	text
toilet paper disp.lctn	sidewall	sidewall, rear wall, front wall	?V = sidewall	text
toilet paper disp. ht. lctn	460	n/a	?V >= 460 AND <= 920	numeric (mm)
toilet paper width lctn	920	n/a	?V <= 920	numeric (mm)

GRAB BAR				
SLOT	**VALUE(S)**	**POSSIBLE-VALUES**	**LEGAL VALUES**	**VALUE TYPE**
s.gb.orient	horizontal	horizontal, verticle, angle	?V = horizontal	text
s.gb.length	157	n/a	?V >= 157	numeric (mm)
s.gb.bar.dia	40	n/a	?V <= 40	numeric (mm)
s.gb.bar.load	1.3	n/a	?V >= 1.3	numeric (kN)
s.gb.bar.clear	40	n/a	?V >=35 AND <= 45	numeric (mm)
s.gb.ht.lctn	880	n/a	?V >= 840 AND <= 920	numeric (mm)
s.gb.extd.fr.lctn	450	n/a	?V >= 450	numeric (mm)
s.gb.extd.back.lctn	450	n/a	?V >= 450	numeric (mm)
rear grab bar	Yes	Yes, No	(?V = Yes AND no tank) OR ?V = No AND tank)	text
rear.gb.width	920	n/a	?V >= 525	numeric (mm)
rear.gb.clear	40	n/a	?V >= 35 and <= 45	numeric (mm)
rear.gb.dia	40	n/a	?V <= 40	numeric (mm)
rear.gb.load	1.3	n/a	?V >= 1.3	numeric (kN)
rear gb.width.lctn	1/2 rear gb width (center on toilet - 300?)	n/a	?V >= 300	numeric (mm)
rear.gb.ht.lctn	880	n/a	?V >= 840 <=920	numeric (mm)

WALL				
SLOT	**VALUE(S)**	**POSSIBLE-VALUES**	**LEGAL VALUES**	**VALUE TYPE**
type	partition	partition, stall partitition, plumbing, loadbearing		list
reinforced or loadbearing	yes	yes, no	?V = yes IF sidewall of toilet	text
type paritition	moveable	moveable, nonmoveable	?V = moveable	text
vestibule	no	yes, no	?V = ???	
vestibule length	2000	n/a	?V >= 2000	numeric (mm)
vestibule width	1500	n/a	(?V >= 1500 AND inswing door) OR (?V >= 1200 AND outswinge door)	numeric (mm)

LAVATORY				
SLOT	**VALUE(S)**	**POSSIBLE-VALUES**	**LEGAL VALUES**	**VALUE TYPE**
type	wall hung	pedestal, wall hung, vanity, countertop	ONEOF (?V, wall hung, countertop)	List
bowl depth	160	n/a	?V <= 160	numeric (mm)
depth lctn	420	n/a	?V >= 420	numeric (mm)
width lctn	460	n/a	?V >= 460	numeric (mm)
top.ht.lctn	865	n/a	?V <= 865 AND >= 735	numeric (mm)
front.ht.clear.lctn	685	n/a	?V >= 685	numeric (mm)
mid.ht.clear.lctn	685	n/a	?V >= 685	numeric (mm)
rear.ht.clear.lctn	230	n/a	?V >= 230	numeric (mm)
clear.depth.lctn	1220	n/a	?V >= 1220 AND <= 460 under sink	numeric (mm)
clear.width.lctn	760	n/a	?V >= 760	numeric (mm)
pipes insulated	No	Yes, No	?V = *	
pipes exposure	No	Yes, No	(?V = Yes AND ?SELF Pipes Insulated = Yes) OR (?V = No)	text
exposed surface	smooth	smooth, sharp, abrasive	?V = smooth	text
faucet type	single lever	single lever, long lever, push button, automatic, daisy, round, short lever	ONE OF (?V, single lever, long lever, push button, automatic)	list
faucet force	5	n/a	?V <5	numeric (lbf)
mirror	yes	yes, no	?V = yes	text
mirror ht. lctn	1000	n/a	?V <= 1000	numeric (mm)
shelf	yes	yes, no	?V = yes	text
shelf ht. lctn	1200	n/a	?V <= 1200	numeric (mm)
soap disp.ht. lctn	1400	n/a	?V <= 1400	numeric (mm)
towel.ht.lctn	1400	n/a	?V <= 1400	numeric (mm)

APPENDIX D

PROTOTYPE II - PROGRAM CODE

```
;
; SIMPLE-FRAME-SYSTEM-WITH-CLASSES
;
; This file contains the set of routines that implement the simple
; frame-based system described in the 74.416 course notes. This is
; an extension to the SIMPLE-FRAME-SYSTEM to handle slot facets and CLASSES!!
;
;
; NOTE that only single inheritance is supported (although any number of levels
; of inheritance can be generated)

; Sample test cases are found at the end of this file
; (just compile the entire file and the test cases will be
;  run automatically)
;

(defvar *CLASS-LIST*)
(defvar *WM*)
(defvar *DEFAULT-SLOT*)

; The first routine we need is MAKE-FRAME-CLASS which simply creates a new frame
; class and inserts it in *CLASS-LIST*
;
;Note that this routine returns the frame class handle
;
;
; Update Subclasses list in p-class

(defun MAKE-FRAME-CLASS (p-class c-name &rest slot-defns)
           (let ((p-class-info (get-frame p-class)))
      (if (or (null p-class) (not (null p-class-info)))
                    (let ((class-handle (gentemp)))
         (set class-handle
                                        (append
                              (list (list 'Id class-handle))
                     (list (list 'Name c-name))
                (list (list 'Superclass p-class))
                slot-defns))
         ;
         ; Update *CLASS-LIST* which holds the list of classes
         ; (class-name class-handle) entries are stored
         ;
         (if (null *CLASS-LIST*)
              (setf  *CLASS-LIST* (list (list c-name class-handle)))
```

```
                (nconc *CLASS-LIST* (list (list c-name class-handle))))

             ;  If a parent class is specified, update its subclass list
             ;
             (if (not (null p-class))
                 (UPDATE-SUBCLASSES p-class class-handle))
             class-handle)
          (progn
            (princ "Invalid Parent Class ... Make-Frame-Class failed!")
            (terpri)
            nil))))

;
;  This routine will update the subclass list associated with a given class
;

(defun UPDATE-SUBCLASSES (class-handle sub-class-handle)
  (let ((subclasses-slot (GET-SLOT class-handle 'Subclasses)))

      (cond ((null subclasses-slot) ; create list and insert 1st subclass
             (INSERT-SLOT class-handle 'Subclasses (list sub-class-handle)))
            (T
             (cond ((null (GET-ACTUAL-VALUE subclasses-slot))
                    (setf (second (GET-SLOT class-handle 'Subclasses))
                          (list sub-class-handle)))
                   (T
                    (nconc
                     (GET-ACTUAL-VALUE (GET-SLOT class-handle 'Subclasses))
                       (list sub-class-handle))))))))

;  Search the *CLASS-LIST* for a given class name ... return the class-handle or
;  nil if the class does not exist

(defun FIND-FRAME-CLASS (c-name)
  (second (assoc c-name *CLASS-LIST*)))

;   Search the instance list *WM* for a given instance name ... return the frame
;   handle or nil if the instance does not exist

(defun FIND-FRAME-INSTANCE (i-name)
  (second (assoc i-name *WM*)))


;The first routine we need is MAKE-FRAME-INSTANCE which creates a new frame and
;inserts it in *WM*
;
;Note that this routine returns the frame handle that was created to act as the
;variable to store the frame
;
;
;  Update INSTANCES list in class

(defun MAKE-FRAME-INSTANCE (c-handle f-name &rest slot-defns)
  (if (and (not (null c-handle))
           (not (null (get-frame c-handle))))
```

```
    (let ((frame-handle (gentemp)))
      (set frame-handle
           (append
            (list (list 'Id frame-handle))
            (list (list 'Name f-name))
            (list (list 'Instance-of  c-handle))
            slot-defns))
      (if (null *WM*)
        (setf  *WM* (list (list f-name frame-handle)))
        (nconc *WM* (list (list f-name frame-handle))))
      (UPDATE-INSTANCES c-handle frame-handle)
      frame-handle)
    (progn
      (princ "Invalid Class .. Make-Frame-Instance failed")
      (terpri)
      nil) ))

;
;  This routine will update the instances list associated with a given class
;

(defun UPDATE-INSTANCES (class-handle instance-handle)
  (let ((instances-slot (GET-SLOT class-handle 'Instances)))

    (cond ((null instances-slot)
           (INSERT-SLOT class-handle 'Instances (list instance-handle)))
          (T
           (cond ((null (GET-ACTUAL-VALUE instances-slot))
                  (setf (second (GET-SLOT class-handle 'Instances))
                        (list  instance-handle)))
                 (T
                  (nconc
                   (GET-ACTUAL-VALUE (GET-SLOT class-handle 'Instances))
                   (list instance-handle))))))))

;Since working memory stores the frame handles as pointers to the individual
;frames, we should create a return the frame structure linked to a given
;frame handle

(defun GET-FRAME (frame-handle)
          (eval frame-handle))


;Next we need a routine that can delete a given class from the class list. We
;will assume that the routine is passed the handle of the class to be deleted.
;
;  We must use remove because DELETE has a bug in the beta version

(defun DESTROY-FRAME-CLASS (class-handle)
  (when (not (null class-handle))
    (setq *CLASS-LIST*
         (remove
          (list
            (GET-SLOT-VALUE class-handle 'name)
            class-handle)
         *CLASS-LIST*
         :test 'EQUAL))

   (MAKUNBOUND class-handle)
   )
  )


;Next we need a routine that can delete a given frame from working memory. We
;will assume that the routine is passed the handle of the frame to be deleted.
;
;  We must use remove because DELETE has a bug in the beta version


(defun DESTROY-FRAME-INSTANCE (frame-handle)
 (let ((parent-class-handle (GET-ACTUAL-VALUE (GET-SLOT frame-handle 'Instance-of)))
       (tmp nil))

   (when (not (null frame-handle))
    (setq *WM*
         (remove
          (list
           (GET-SLOT-VALUE frame-handle 'name)
           frame-handle)
          *WM*
          :test 'EQUAL))

    (MAKUNBOUND frame-handle)
    )

   (setq tmp (remove frame-handle
                (GET-ACTUAL-VALUE (GET-SLOT parent-class-handle 'Instances))
                :test 'equal))
   (setf (second (GET-SLOT parent-class-handle 'Instances)) tmp)
   )
 )



;Now that we can create, delete, and modify frames, we need a set of routines
;to print an individual frame and all frames currently stored in WM
;
; note that elements of *WM* are (instance-name instance-handle)


(defun PRINT-WM ()
        (princ "The frames currently in *WM* are:")
        (terpri)
        (dolist (frame-handle-info *WM*)
                (PRINT-FRAME   (second frame-handle-info))
                (terpri)) )

; This routine prints an individual frame given a
; frame id (pointer)
```

```
(defun PRINT-FRAME (frame-handle)
        (let ((frame (GET-FRAME frame-handle)))
      (when (not (null frame))
       (princ (GET-SLOT-VALUE frame-handle 'name))
       (princ " (")
       (princ frame-handle)
       (princ ")")
       (terpri)
       (dolist (slot (cddr frame))
        (format t "    ~15A:  ~A" (first slot)
            (cond ((equal (first slot) 'Instance-of)
                (GET-SLOT-VALUE (GET-ACTUAL-VALUE slot) 'Name))
               (t (second slot))))
        (terpri))
       (terpri) ) ) )

(defun PRINT-CLASSES ()
        (princ "The classes currently defined are:")
        (terpri)
        (dolist (class-handle-info *class-list*)
                (PRINT-CLASS   (second class-handle-info))
                (terpri)) )

; This routine prints an individual class given a
; class handle

(defun PRINT-CLASS (class-handle)
 (let ((frame (GET-FRAME class-handle)))
  (when (not (null frame))
   (princ (GET-SLOT-VALUE class-handle 'name))
   (princ " (")
   (princ class-handle)
   (princ ")")
   (terpri)
   (dolist (slot (cddr frame))
    (format t "    ~15A:  " (first slot))
    (cond ((equal (first slot) 'Superclass)
        (princ (GET-SLOT-VALUE  (GET-ACTUAL-VALUE slot) 'Name)))
       ((equal (first slot) 'Subclasses)
        (print-frame-names slot))
       ((equal (first slot) 'Instances)
        (print-frame-names slot))
       (t (princ (second slot))))
    (if (third slot)
     (progn
      (terpri)
      (princ "                    ")
      (princ (third slot))))
    (terpri))
  (terpri) ) ) )


(defun PRINT-FRAME-NAMES (slot)
 (princ "( ")
 (dolist (handle (GET-ACTUAL-VALUE slot))
   (if (not (null (GET-FRAME handle)))
       (progn
         (princ (GET-SLOT-VALUE handle 'Name))
         (princ " "))))
  (princ ")"))

; The following routines operate on a given frame handle (frame-instance
; or class)


; This routine returns a slot from a given frame
;
; NIL will be returned if the slot is not found in the frame

(defun GET-SLOT (frame-handle slot)
  (let ((frame (GET-FRAME frame-handle)))
                   (assoc slot frame :test 'equal)))


;
;  The following routine accesses the value component of a given slot
;  The slot must be a list. This routine does not support inheritance
;   see GET-SLOT-VALUE defined below.

(defun GET-ACTUAL-VALUE (slot)
  (second slot))


; This routine returns the facet list for a given slot
;
;  If not found it returns NIL
;
; No inheritance is used

(defun GET-FACET (slot)
  (third slot))

; The following routine will return the contents of a given facet of a given
; slot structure. The slot structure is assumed to be non-nil. If the given
; facet is not found in the slot structure then return NIL (the DEFAULT SLOT
; info is not used since inheritance may be employed by the calling routine)


(defun GET-SLOT-FACET (slot facet)
  (let* ((facet-list (GET-FACET slot))
         (facet-info (assoc facet facet-list)))

       (if (or (null facet-list) (null facet-info))
          nil
          (second facet-info))))


;
; Determine the parent class of a given frame

(defun PARENT-OF (frame-handle)
  (let ((instance-of (GET-SLOT frame-handle 'Instance-of))
        (superclass (GET-SLOT frame-handle 'Superclass)))

       (cond ((not (null instance-of))
              (GET-ACTUAL-VALUE instance-of))
             ((not (null superclass))
              (GET-ACTUAL-VALUE superclass))
             (t nil))))


; This routine will return the value associated with a given facet of a given
; slot. If the slot is not found in the frame then inheritance is used
; IF all else fails then the *DEFAULT-SLOT* info is used


(defun GET-SLOT-FACET-SPECIAL (frame-handle slot facet-type)
  (let ((slot-info (GET-SLOT frame-handle slot))
        (value nil))

       (if (not (null slot-info))
           (setq value (GET-SLOT-FACET slot-info facet-type)))

       (if (null value)
           (let ((class-handle (PARENT-OF frame-handle)))
                (if (not (null class-handle))
                    (GET-SLOT-FACET-SPECIAL class-handle slot facet-type)
                    (GET-SLOT-FACET *DEFAULT-SLOT* facet-type)))
           value)))


; This routine will return the value associated with a given facet of a given
; slot. If the slot is not found in the frame then inheritance is used
; IF all else fails then the *DEFAULT-SLOT* info is used


(defun GET-SLOT-VALUE-SPECIAL (frame-handle slot)
  (let ((slot-info (GET-SLOT frame-handle slot))
        (value nil))

       (if (not (null slot-info))
           (setq value (GET-ACTUAL-VALUE slot-info)))

       (if (null value)
           (let ((class-handle (PARENT-OF frame-handle)))
                (if (not (null class-handle))
                    (GET-SLOT-VALUE-SPECIAL class-handle slot)
                    (GET-ACTUAL-VALUE *DEFAULT-SLOT*)))
           value)))

;
;  This routine attempts to retrieve the value associated with a given slot
;  of a given frame. If the value is not found in the current frame then
;  inheritance is used. If this fails then the when-needed routine is employed
;  Not that DEPTH-FIRST search is used (all slots up the hierarchy are searched
;  before the when-needed routine(s) are used.  The *DEFAULT-SLOT* value must
;  be NIL or UNKNOWN or the when-needed routine option will never be used.
```

```
(defun GET-SLOT-VALUE (frame-handle slot)
  (let ((value (GET-SLOT-VALUE-SPECIAL frame-handle slot)))

     (if  (not (equal value 'UNKNOWN))
          value
          (ACTIVATE-WHEN-NEEDED frame-handle slot))))

;
; The following routine will attempt to invoke the when-needed routine
; associated with agiven slot of a given frame. It should be called
; by the GET-SLOT-VALUE routine
;
;
; The when-needed routine should handle whether the slot's value should be
; updated!

(defun ACTIVATE-WHEN-NEEDED (frame-handle slot)
  (let ((slot-info (GET-SLOT frame-handle slot))
       (when-needed-routine nil))

     (when (not (null slot-info))
          (setq when-needed-routine
               (GET-SLOT-FACET-SPECIAL frame-handle slot 'when-needed))
          (if (not (null when-needed-routine))
              (funcall when-needed-routine frame-handle slot)))
     ))


; This routine will attempt to invoke the when-changed routine
; associated with a given slot. It should be called by SET-SLOT-VALUE
;
; It passes the slot name and frame-handle to the routine found in
; in the slot's when-changed facet

(defun ACTIVATE-WHEN-CHANGED (frame-handle slot)
 (let ((slot-info (GET-SLOT frame-handle slot))
       (when-changed-routine nil))

     (when (not (null slot-info))
          (setq when-changed-routine
               (GET-SLOT-FACET-SPECIAL frame-handle slot 'when-changed))
          (if (not (null when-changed-routine))
              (funcall when-changed-routine frame-handle slot)))
     ))


; This routine inserts a new slot into an existing frame
;
; We assume that the slot does not already exist
;
; The frame must exist!!
;
; Facet list is optional

(defun INSERT-SLOT (frame-handle slot value &optional facet-list)
(let ((frame (GET-FRAME frame-handle)))
    (if (not (null frame))
       (if (null facet-list)
                          (nconc frame  (list (list slot value)))
           (nconc frame  (list (list slot value facet-list)))))

    value))



;
; This routine sets the given slot of a given fram to the given
; value. It calls insert-slot if the given slot is not found
; in the frame

(defun SET-SLOT-VALUE (frame-handle slot value)
                    (let ((slot-info (GET-SLOT frame-handle slot)) )
      (cond ((null slot-info)
                            (INSERT-SLOT frame-handle slot value))
          (T
           (setf (second (get-slot frame-handle slot))
               value)))
      (activate-when-changed frame-handle slot)))



; The following routine handles the message passing capabilities of the
; frame system. A frame can have a slot that contains a function pointer
; or actual function definition. The slot type facet should be set to METHOD
; so that the system knows that this is a method and not data.
;
; The SEND function simply takes a frame-handle and a method name and
; simulates sending a message to the frame to invoke the function in
; the slot corresponding to the method name. We still want to use inheritance
; so we process the request as if we were asked to retrieve the value
; from the slot. Then we simply invoke the value (which
; should be checked first to ensure it is a method). We will assume that
; all methods accept a frame-handle as their first argument. Any other arguments
; can be user-defined

(defun SEND (frame-handle method-name &rest args)
 (if (equal (GET-SLOT-FACET-SPECIAL frame-handle method-name 'type) 'METHOD)
    (let ((method (GET-SLOT-VALUE frame-handle method-name)))

       (if (and  (not (null method)) (not (equal method 'UNKNOWN)))
          (apply method (cons frame-handle args))
          (progn
           (princ "Invalid frame or method name")
           (terpri)
           nil)))
    (progn
     (princ "Invalid method ... slot contains data")
     (terpri)
     nil)))
```

```
; The following routines handle the translation of slot constraints for
; evaluation
; ----------------------------------------------------------------------
;
;  Perform a translation on a token of the form [SELF].attr
;
(defun TRANSLATE-SELF-ATTR (token frame-handle)
 (and (valid-token token)
     (let* ((tmp-token (string-upcase (convert-to-string token) ':end nil))
           (index (string-position tmp-token "[SELF]."))
           (attr (and index (subseq tmp-token (+ index 7)))))

          (if attr
            (list 'GET-SLOT-VALUE
                (list 'quote frame-handle)   ; handle to SELF
                (list 'quote (intern attr)))))))


;
;  Perform a translation on a token of the form [SELF.attr1].attr2
;
(defun TRANSLATE-SELF-ATTR-INDIRECT (token frame-handle)
 (and (valid-token token)
     (let* ((tmp-token (string-upcase (convert-to-string token) ':end nil))
           (index1 (string-position  tmp-token "[SELF."))
           (index2 (string-position  tmp-token "]."))
           (attr1 (and index1 index2
                  (subseq tmp-token (+ index1 6)  index2)))
           (attr2 (and index1 index2 (subseq tmp-token (+ index2 2)))))

          (if (and attr1 attr2)
            (list 'GET-SLOT-VALUE
                (list
                 'quote
                 (FIND-FRAME-INSTANCE  ; access instance based on
                                ; instance-name in slot
                    (GET-SLOT-VALUE frame-handle (intern attr1))))
                (list 'quote (intern attr2)))))))


;  This routine handles the ?v token which is translated to a call to get
;  the value associated with the current slot (possibly by searching up the
;  hierarchy or using an if-needed method)

(defun TRANSLATE-?V (token frame-handle slot)
 (if (equal '?V token)
    (list 'GET-SLOT-VALUE
        (list 'quote frame-handle)    ; handle to SELF
        (list 'quote slot))))



; The translation process for the constraint expression in a slot
; is as follows (to create an executable version of the expresion):
```

```
;
; for each list (starting at the top level):
;
;   leave the first element in the list as is (this must be a valid function)
;   translate all other elements as follows:
;
;     ?v  -  (get-slot-value 'frame-handle 'slot)
;            note that frame-handle is the handle of the current object and
;            SLOT is the current slot name
;
;     [self].attr - (get-slot-value 'frame-handle 'attr)
;            note that frame-handle is the handle of the current object
;
;     [self.attr1].attr2  - (get-slot-value 'frame-handle 'attr2)
;
;            note that frame-handle is the handle of the instance obtained
;            by using the value stored in attr1 of the current object
;            this allows an expression to refer to other objects and have
;            the referal resolved at runtime (the appropriate slot in the
;            object should have the name of the instance to refer to)
;
;     all other symbols are translated to 'symbol
;
;
; The translated expression can then be evaluated by doing an (EVAL expr)
;
;
; Note that functions must be represented in brackets - even ones that do not
; have arguments
;
;  optimized to use NCONC rather than append if expression has greater than
;  2 elements
;
;  allows nil to be used in an expression ... translates to 'nil properly

(defun TRANSLATE-EXPRESSION (expr frame-handle slot)
 (cons
  (first expr)
  (let* ((second-arg (second expr))
         (arg-list (and (> (length expr) 1)
                        (list (TRANSLATE-ARGUMENT
                               second-arg frame-handle slot)))))

    (dolist (arg (rest (rest expr)))
     (nconc arg-list
          (list (TRANSLATE-ARGUMENT arg frame-handle slot)))
     )
    arg-list)))

(defun TRANSLATE-ARGUMENT (arg frame-handle slot)
 (if (and arg (listp arg) (equal (first arg) 'quote))
   arg  ; a quote list should be left as is
   (if (and arg (listp arg))  ; list (not nil) .. recursive translation applies

     (TRANSLATE-EXPRESSION arg frame-handle slot)
```

```
      (or  ; atom ... must translate according to rules
       (TRANSLATE-SELF-ATTR arg frame-handle)
       (TRANSLATE-SELF-ATTR-INDIRECT arg frame-handle)
       (TRANSLATE-?V arg frame-handle slot)
       (list 'quote arg)
       )
      ))
  )


; The following routines handle the translation of slot constraints for
; explanation of constraints to the user (result cannot be evaluated by LISP)
; ----------------------------------------------------------------------
;
;  Perform a translation on a token of the form [SELF].attr
;  to the string [OBJECT-NAME.attr]
;
(defun TRANSLATE-SELF-ATTR-EXPL (token frame-handle)
 (and (valid-token token)
      (let* ((tmp-token (string-upcase (convert-to-string token) ':end nil))
             (index (string-position tmp-token "[SELF]."))
             (attr (and index (subseq tmp-token (+ index 7)))))

            (if attr
               (format nil "[~S.~A]"
                      (GET-SLOT-VALUE frame-handle 'name)
                      attr)))))


;
;  Perform a translation on a token of the form [SELF.attr1].attr2
;  to the string [OBJECT-NAME.attr2]
;
(defun TRANSLATE-SELF-ATTR-INDIRECT-EXPL (token frame-handle)
 (and (valid-token token)
      (let* ((tmp-token (string-upcase (convert-to-string token) ':end nil))
             (index1 (string-position  tmp-token "[SELF."))
             (index2 (string-position  tmp-token "]."))
             (attr1 (and index1 index2
                         (subseq tmp-token (+ index1 6)  index2)))
             (attr2 (and index1 index2 (subseq tmp-token (+ index2 2)))))

            (if (and attr1 attr2)
               (format nil "[~S.~A]"
                      (GET-SLOT-VALUE frame-handle (intern attr1))
                      attr2)))))


; The translation process for the constraint expression in a slot
; is as follows (to create an explanation string only):
;
; for each list (starting at the top level):
;
;   leave the first element in the list as is
;   translate all other elements as follows:
```

```
;
;    ?v  -  stays as ?v
;
;    [self].attr - "[object-name.attr]"
;
;
;    [self.attr1].attr2  - "[object-name.attr2]"
;
;
;
;    all other symbols are left as is
;
;
; The translated expression can NOT be evaluated
;
;
;


(defun TRANSLATE-EXPRESSION-EXPL (expr frame-handle slot)
 (if (null expr)
    nil
    (cons
      (first expr)
      (let* ((second-arg (second expr))
             (arg-list (and (> (length expr) 1)
                            (list (TRANSLATE-ARGUMENT-EXPL
                                   second-arg frame-handle slot)))))

          (dolist (arg (rest (rest expr)))
                 (nconc arg-list
                        (list (TRANSLATE-ARGUMENT-EXPL
                                   arg frame-handle slot)))
                 )
          arg-list))))

(defun TRANSLATE-ARGUMENT-EXPL (arg frame-handle slot)
 (if (and arg (listp arg))  ; list (not nil) .. recursive translation applies

    (TRANSLATE-EXPRESSION-EXPL arg frame-handle slot)

    (or (TRANSLATE-SELF-ATTR-EXPL arg frame-handle)
        (TRANSLATE-SELF-ATTR-INDIRECT-EXPL arg frame-handle)
        arg)))



;
;  The following routine will accept an explanation version of a constraint
;  and it will build the footnote list that contains strings of the form
;  "[object-name].attr  = VALUE"

(defun BUILD-EXPL-FOOTNOTE1 (object-handle expl-list)
 (let ((footnote nil))
    (dolist (elem expl-list)
          (if (and elem (listp elem))
              (setq footnote
                   (append footnote
                         (BUILD-EXPL-FOOTNOTE1 object-handle elem)))
              (if (and (stringp elem)
                     (string-position elem "[SELF]."))
                 (setq footnote
                      (append footnote
                            (list
                             (concatenate  'string
                                 (string-upcase elem)
                                 " = "
                                 (string
                                  (GET-OBJECT-VALUE-EXPL1 object-handle elem)))))))
              ))
      footnote))

;
; The following routine will accept an explanation version of a constraint
; and it will build the footnote list that contains strings of the form
; "[object-name.attr] = VALUE"

(defun BUILD-EXPL-FOOTNOTE2 (expl-list)
 (let ((footnote nil))
      (dolist (elem expl-list)
           (if (and elem (listp elem))
              (setq footnote
                   (append footnote
                         (BUILD-EXPL-FOOTNOTE2 elem)))
              (if (and (stringp elem)
                     (equal (aref elem 0) #\[))
                 (setq footnote
                      (append footnote
                            (list
                             (concatenate  'string
                                 (string-upcase elem)
                                 " = "
                                 (convert-to-string
                                  (GET-OBJECT-VALUE-EXPL2 elem)))))))
              ))
      footnote))


(defun GET-OBJECT-VALUE-EXPL1  (object-handle str)
 (let* ((tmp-str (string-upcase str))
        (pos-of-dot (string-position tmp-str "."))
        (pos-of-]   (string-position tmp-str "]"))
        (attr-name (and pos-of-] pos-of-dot
                      (subseq tmp-str (1+ pos-of-dot)))))
        (GET-SLOT-VALUE
            object-handle
            (intern attr-name))))


(defun GET-OBJECT-VALUE-EXPL2  (str)
 (let* ((tmp-str (string-upcase str))
        (pos-of-dot (string-position tmp-str "."))
        (pos-of-]   (string-position tmp-str "]"))
        (object-name (and pos-of-dot
                          (subseq tmp-str 1 pos-of-dot)))
        (attr-name (and object-name
                        pos-of-]
                        (subseq tmp-str (1+ pos-of-dot) pos-of-]))))
        (GET-SLOT-VALUE
           (FIND-FRAME-INSTANCE (intern object-name))
           (intern attr-name))))


; This routine will print out the constraint expression in a slot if
; slot contains a constraint.
; A footnote of the values of object.attr references will also be printed.
; This is an explanation facility
;
; should add an optional parameter to indicate whether the constraint is
; satisfied (must search constriant violation list) *****

(defun PRINT-SLOT-CONSTRAINT-EXPL (frame-handle slot &optional (show-status))
  (let ((constraint-expl (TRANSLATE-EXPRESSION-EXPL
                          (GET-SLOT-FACET-SPECIAL frame-handle
                                                  slot 'legal-values)
                          frame-handle slot)))
      (if (not (null constraint-expl))
         (progn
          (terpri)
          (princ "  ")
          (princ slot)
          (princ ":  ")
          (princ "(?V = ")
          (princ (GET-SLOT-VALUE frame-handle slot))
          (princ ")")
          (terpri)
          (princ "      ")
          (princ constraint-expl)
          (terpri)
          (dolist (note (BUILD-EXPL-FOOTNOTE1 frame-handle constraint-expl))
            (terpri)
            (princ "          ")
            (princ note)
            )
          (dolist (note (BUILD-EXPL-FOOTNOTE2 constraint-expl))
            (terpri)
            (princ "          ")
            (princ note)
            )

          ))))


;  print out constraint info for all slots of a given frame (those that
;  have constraints defined)

(defun PRINT-FRAME-CONSTRAINT-EXPL (frame-handle)
  (print (GET-SLOT-VALUE frame-handle 'name))
  (dolist (slot (GET-FRAME frame-handle))
       (PRINT-SLOT-CONSTRAINT-EXPL frame-handle (first slot)))
  (terpri)
  )



; The following routine will apply the constraint facet (legal-values) defined
; for each slot in a given frame instance. This routine is only applicable to
; frame instances at this time (because indirect references assume that other
; instances are used (should probably support either .. check for instance
;  and if this fails check for class)
;
;  returns a violations list ((slot name constraint-list) ... )

(defun APPLY-CONSTRAINTS-IN-FRAME (frame-handle)
 (let (results
      retcode
      slot-name)

     (dolist (slot-info (GET-FRAME frame-handle))
          (setq slot-name (first slot-info))
          (setq retcode
               (APPLY-CONSTRAINTS-IN-SLOT
                     frame-handle
                     slot-name))
          (if (not retcode)
             (setq results
                 (append results
                   (list  slot-name))))
          )
     results))


;
;  apply constraints in a given slot of a given frame
;
;  (frame instances only for now)
;
;
;  **** ADD facility to check for UNKNOWN or NIL as the slot value to return
;  no value or valid

(defun APPLY-CONSTRAINTS-IN-SLOT (frame-handle slot)
 (let* ((expr
        (GET-SLOT-FACET-SPECIAL frame-handle slot 'legal-values))
       (translated-expr
        (and expr
            (TRANSLATE-EXPRESSION expr frame-handle slot)))
       (result
        (and translated-expr
            (eval translated-expr))))

  (cond ((null expr)
        ;"No constraints"
        T)
        ((null result)
        ;"Constraint Violation"
        nil)
        (t
        ;"Constraint(s) satisfied
        T))
  ))


; check a given list of frame-handles for constraint violations

(defun CHECK-FRAME-LIST (f-list)
  (let (constraint-violations
        results)
       (dolist (f-info f-list)
            (setq constraint-violations
                 (APPLY-CONSTRAINTS-IN-FRAME (second f-info)))
            (if constraint-violations  ; at least one violation
               (setq results
                    (append results
                         (list (list (first f-info)
                                   constraint-violations)))))
            )
       results))


(defun CHECK-OBJECTS ()
  (let ((problems (CHECK-FRAME-LIST *WM*)))
    (if (null problems)
      (princ "*** No problems found ***")
      (PRINT-PROBLEMS problems) )
    (WRITE-PROBLEMS-TO-DISK problems)
    (WRITE-OBJECT-TO-DISK problems)
    )
  (terpri)
  )


;
;
;  writes first obect it finds with a problem to VIOLATION file... this is what works with
;  the current version of SHOW VIOLATIONS in the MINICAD tool

(defun WRITE-OBJECT-TO-DISK (problems)
  (if (not (null problems))
    (let ((outfile (open "HardDisk:AI applications:cad:MiniCAD+3.8:Violation" :direction :output
                    :if-does-not-exist :create :if-exists :rename-and-delete)))
      (princ (first (first problems)) outfile)
      (close outfile)))
  )

;
;
; Writes out a detailed file describing object errors found
;
```

```
; Object1-id
; number-of-attribute-errors
; attr1
; value
; error-string
; attr2
; value
; error-string
; Object2-id
; number-of-attributes-errors
; attr1
; value
; error-string
; attr2
; value
; error-string
; ...

(defun WRITE-PROBLEMS-TO-DISK (problems)
 (let ((outfile (open "HardDisk:AI Applications:cad:MiniCAD+3.8:Violation-Info" :direction :output
                 :if-does-not-exist :create :if-exists :rename-and-delete))
     ;(num-problems (length problems))
     )

   ; now print out the actual problem details (if any)
   (dolist (p problems)
    (let ((object (first p)))
     (prin1 object outfile)
     (terpri outfile)
     (prin1 (length (second p)) outfile)  ; number of attributes with errors in the object
     (terpri outfile)
     (dolist (slot (second p))  ; loop thru attribute error list
      (PRINT-SLOT-CONSTRAINT-EXPL-STRING-TO-DISK (FIND-FRAME-INSTANCE object) slot outfile)
      )
     )
    )
   (close outfile)))


;
;
; print slot value error-string to disk
;

(defun PRINT-SLOT-CONSTRAINT-EXPL-STRING-TO-DISK (frame-handle slot outfile)
 (prin1 slot outfile)
 (terpri outfile)
 (prin1 (GET-SLOT-VALUE frame-handle slot) outfile)
 (terpri outfile)
 (let ((expl (GET-SLOT-FACET-SPECIAL frame-handle slot 'explain)))
  (dolist (x expl)
   (princ x outfile)
   (princ " " outfile)
   )
  (terpri outfile)
 ))
```

```
;
;
; prints problems found to lisp listener window
;

(defun PRINT-PROBLEMS (problems)
 (terpri)
 (princ "The following Constraint Violations were Detected:")
 (terpri)
 (terpri)
 (dolist (p problems)
  (let ((object (first p)))
   (princ "Object: ")
   (princ object)
   (terpri)
   (dolist (slot (second p))
   ; (PRINT-SLOT-CONSTRAINT-EXPL (FIND-FRAME-INSTANCE object) slot)
    (PRINT-SLOT-CONSTRAINT-EXPL-STRING (FIND-FRAME-INSTANCE object) slot)
    )
   (terpri)
   )
  )
 )


(defun PRINT-SLOT-CONSTRAINT-EXPL-STRING (frame-handle slot)
 (terpri)
 (princ " ")
 (princ slot)
 (princ ":  ")
 (princ "(")
 (princ (GET-SLOT-VALUE frame-handle slot))
 (princ ")")
 (terpri)
 (princ "   ")
 (let ((expl (GET-SLOT-FACET-SPECIAL frame-handle slot 'explain)))
  (dolist (x expl)
   (princ x)
   (princ " ")
   )
  (terpri))
 )

; The following routine will initialize the frame system. For now all we
; need to do is set WM to NIL inidicating that no frames currently exist.

(defun init-frame-system ()
 (setf *WM* nil)
 (setf *CLASS-LIST* nil)
 (setf *DEFAULT-SLOT* '(DEFAULT-SLOT-INFO
                        UNKNOWN
                        (
                         (legal-values nil)
                         (type DATA)
                         (constraints nil)
```

```
                    (when-changed nil)
                    (when-needed nil)
                    (explain ("A violation has occurred"))
                    (inherit override))))
 (terpri)
 (format t "Frame system initialized ...")
 (terpri))



;
;  We only need to search symbols for special symbols like [SELF]

(defun VALID-TOKEN (token) (or (stringp token) (symbolp token)))

;
;
;
;
;
;  Routines that maintain compatibility between ExperCommonLisp and MacLisp
;
;
;
;
; compile the next routine for MacLisp Only

(defun STRING-POSITION (str1 str2)
 (search str2 str1))

(defun CONVERT-TO-STRING (x)
 (format nil "~S" x))



;
;
;
(defvar root-class)
(defvar door-class)
(defvar room-class)
(defvar toilet-class)
(defvar object-1)

(defun DEFINE-KB ()
 (init-frame-system)

 (setq root-class
     (make-frame-class  nil 'ROOT
                  '(Describe print-frame ((type method)))))


 (setq door-class
     (make-frame-class root-class 'DOOR
                 '(Door-Width 800
                     ((legal-values
                       (OR (> ?V 800)
                    (AND (> ?V 760)
                         (EQUAL [SELF.room].type Residential))))
               (explain
                ("The door must be at least 800mm wide."
                 "The width can be reduced to"
                 "760mm for residential washrooms."))
               ))
         '(Room WASH-ROOM1)
         '(knob-type lever
               ((legal-values
                (member ?v '(lever pushplate panic-bar D-pull)))
               (explain
                ("The door knob must be a lever, pushplate,"
                 "panic bar or D-pull for handicap accessible"
                 "washrooms."))
               ))
         '(pull-length 140
               ((legal-values
                (>= ?v 140))
               (explain
                ("The length of the door pull must be at least 140mm."))
               ))
         '(lock-type slider
               ((legal-values
                (member ?v '(graspable-latch flip slider)))
               (explain
                ("The lock must be a graspable latch,"
                 "or a flip or slider type."))
               ))
         '(coathook yes
               ((legal-values
                (equal ?v yes))
               (explain
                ("A coat hook must be present."))
               ))
         ))

(setq room-class
   (make-frame-class root-class 'ROOM
         '(Type Residential)
         ))

(setq toilet-class
   (make-frame-class root-class 'TOILET
         '(Bowl-width 490
               ((legal-values
                (AND (>= ?v 460) (<= ?v 525)))
               (explain
                ("The width of the toilet bowl must be"
                 "between 460mm and 525mm."))
               ))
         '(Back-support yes
               ((legal-values
                (equal ?v yes))
               (explain
                ("The toilet must have a back support."))
```

```
                        ))
                  '(control-lctn openside-tank
                        ((legal-values
                          (member ?v '(openside-tank openside-wall closeside-wall
                                closeside-tank floor)))
                         (explain
                          ("The flush control for the toilet must be "
                           "located on the floor, on the tank or on the wall."))
                         ))

                  ))

 )


;
;  load objects from mini-cad output file
;
(defun CHECK-DESIGN ()
 (CLEAR-OBJECTS)

 ;
 ; we need a default washroom object

 (setq object-1
         (make-frame-instance room-class 'WASH-ROOM1))
 (LOAD-OBJECTS-FROM-FILE "HardDisk:AI Applications:cad:MiniCAD+3.8:MarkInput")
 (terpri)
 (terpri)
 (princ "Checking the Design ...")
 (terpri)
 (terpri)
 (CHECK-OBJECTS)
 )

(defun CLEAR-OBJECTS ()
 (dolist (object *WM*)
  (DESTROY-FRAME-INSTANCE (second object))
  )
 (setq *WM* nil))



;
;  load the knowledge base
;
;(DEFINE-KB)




(defun MOVE-AGENT (frame-handle direction)
 (print direction)
 (set-slot-value frame-handle 'location '(1 1)))
```

```
(setq i-list nil)

(defun issue-instruction (instr-name &rest args)
 (print instr-name)
 (print args)
 (setf i-list (append i-list (list (cons instr-name args))))
 )

(defun execute-instructions (frame-handle i-list)
 (dolist (instr i-list)
  (apply 'send (cons frame-handle instr))
  )
 )

(init-frame-system)

(setq root-class
     (make-frame-class  nil 'ROOT
                  '(Describe print-frame ((type method)))))

(setq primary-agent
    (make-frame-class root-class 'PRIMARY-AGENT
                '(name nil)
                '(move-agent MOVE-AGENT ((type method)))
                '(location (0 0))
                ))

(setf agent-1
    (make-frame-class primary-agent 'agent-1
                '(name BILL)))


(issue-instruction 'move-agent 'south)

(execute-instructions agent-1 i-list)

(print-frame agent-1)
```

ADDITIONAL REFERENCES

Al-Harkan, A. 199_. "An intelligent environment for design evaluation and feedback: a machine learning approach." (doctoral proposal submission). Ann Arbor, Mich.: University of Michigan.

Anders, J.K. 1990. "Through the aperture: an integrated design and database environment." *Macintosh-Aided Design.* July. 54 - 59.

Armstrong, M., S. De, P. Densham, P. Lolonis, G. Rushton, V. Tewari. 1990. "A knowledge-based approach for supporting locational decision-making." *Environment and Planning B.* 17. Great Britain: Pion. 341-364.

Autran, J., Guerin-Cazorla, T. Prevost, P. Saunier. 1991. "A computer system for rehabilitation of buildings in their urban setting." *Environment and Planning B.* 18. Great Britain: Pion. 99 - 106.

Beaumont, C.E. 1992. "Making design review boards work." *Architectural Record.* 180 (1) S. Kliment (ed.) New York: McGraw-Hill Inc. 34 & 54.

Butt, T.K. 1989. "How CADD helped restorations." *Architecture.* (78) November. 115 - 118.

Cao, X., H. Zhijun, Y. Pan. 1990. "Automated design of house-floor layout with distributed planning." *Computer Aided Design.* 22 (4) May. 213 - 221.

Cho, M.S. 1989. "Automated learning of programmatic information from plans of existing buildings." (doctoral dissertation) University of Michigan.

City of Winnipeg. 1986. *Design Guidelines -- Historic Winnipeg Restoration Area.* Winnipeg: Dept of Environmental Planning.

Coyne, R.and S. Newton. 1990. "Design reasoning by association." *Environment and Planning B.* 17. Great Britain: Pion.

Cuandrench, F.C. 1991. "Treatment of knowledge and its use in construction applicaitons -- state of the art and results of a case study." *VTT Symposium -- Computers and Building Regulations.* 27 - 29 May 1991. Espoo, Finland: Technical Reserach Centre of Finland. 53 - 64.

Dave, B., G. Schmitt, B. Faltings, I. Smith. 1994. "Case based design in architecture." *Artificial Intelligence in Design.* The Netherlands: Kluwer Academic Publishers. 145 - 162.

Del Sesto, C. "Computers: electronic code information." *Progressive Architecture.* 71 (7) New York: McGraw-Hill Inc. 52.

Flemming,U. 1986. "The role of shape grammers in the analysis and creation of designs." *The Computability of Design - 1986 Suny Buffalo Symposium on CAD.* A.C. Harfman, Y. Kalay, B.R. Majkowski, L.M. Swerdloff (eds.) Buffalo: State University of New York. Part 2

Fram, M. 1988. *Well-Preserved -- The Ontario Heritage Foundations Manual of Principles and Practice for Architectural Conservation.* Ontario: Boston Mills Press.

Frye, M.J., D.M. Olynick, R.B. Pinkey. 1992. "The development of an expert system for the fire protection requirements of the National Building Code of Canada." presented at *Joint CIB "Integrated Computer Aided Design" and "Computers and Building Standards" Workshops.* 12-14 May 1992. Montreal, Canada.

Gatton, T.M. and F.W. Kearney. 1988. "Expert Systems: automated knowledge base development through graphical modeling." *ARCC Research Conference Conference '88.* School of Architecture, University of Illinois, Urbana.

Gardan,Y. and M. Lucas. 1984. "Interactive graphics in CAD." New York: UNIPUB.

Gero, J.S. and A.D. Radford. 1988. *Design by Optimization in Architecture, Building and Construction.* New York: Van Nostrand Reinhold Company.

Gero, J.S. 1973. "Application of sequential decision-making in optimization problems in architecture." *Computer Report .* 23. Sydney: University of Sydney.

Gero, J. and R. Coyne. 1984. "The place of expert systems in architecture." *CAD84 - 6th International Conference on Computers and Design Engineering -- Journal Computer Aided Design .* J. Wexler (ed.) UK: Butterworths. 529 - 546

Han, S.Y, T.J. Kim, I. Adiguzel. 1991. "XPlanner: A knowlege-based decision support system for facility management and planning." *Environment and Planning B.* 18. Great Britain: Pion. 205 - 224.

Harfman, A.C. and B.R. Majkowski. 1992. "Component-based spatial reasoning." *Computer Supported Design in Architecture - Mission , Method, Madness ACADIA 1992 Proceedings.* K. Kensek and D. Nobel (eds.) University of Southern California.

Harfman, A.C. and S.S. Chen. 1990. "Component-based building representation." *Representation and Simulation in Architectural Research and Design.* 31 March - April 1. Robinson (ed.) State University of New York at Buffalo.

Hattis, D.B. 1981. "How existing buildings and building rehabilitation are regulated." *Association of Preservation Technology Bulletin.* XIII (2). 9 - 12.

Hayes-Roth, F., D. Waterman, D. Lenat. *Building Expert Systems.* Massachusetts: Addison-Wesley Publishing Co. Ltd.

Hedge, A. and D.S. Ellis. 1991. "New graphic database software for managing facilities performance." *IFMA Fifth Annual FACILITIES '91: Computer-Aided FM and High-Tech Systems Conference.* Washington, D.C.: International Facility Management Association. 64-77

Heikkila, E.J. and E.J. Blewett. 1992. "Using expert systems to check compliance with municipal building codes." *Journal of the American Planning Association.* (58) Winter. 72 - 80.

Kennedy, C.B. 1990. "Videographic documentation, analysis, and manipulation -- A new tool for cultural resource management." *Association for Preservation Technology Bulletin.* XXII (1/2). 97 - 103.

Kreider, J.K. and W.H. Wubbena. 1991. "Expert systems for facilty management in commercial buildings." *IFMA Fifth Annual FACILITIES '91: Computer-Aided FM and High-Tech Systems Conference.* Washington, D.C.: International Facility Management Association. 79 - 93

Lee, Y.C. 1990. "Geographic Information Systems for urban applications: problems and solutions." *Environment and Planning B: Planning and Design.* 17. Great Britain: Pion. 463 - 473.

Leishman, D. 1991. "History reuse in building design." (doctoral dissertation) University of Calgary.

Manitoba Culture, Heritage, and Recreation. 1990. "Guidelines for conserving our heritage buildings." Winnipeg: Province of Manitoba.

McIntosh, P.G. 1986. "Models of spatial information in computer-aided design: a comparative study." *The Computability of Design - 1986 Suny Buffalo Symposium on CAD.* A.C. Harfman, Y. Kalay, B.R. Majkowski, L.M. Swerdloff (eds.) Buffalo: State University of New York. Part 2.

McLeod, D. 1992. "Architecture and databases." *Canadian Architect.* 24 - 27.

McNeil, E.S., and B. Scott. 1990. "PREServe -- A geographic information system for reording, analyzing and managing historic resrouces." (research project) Mother Lode Spatial and Architectural Research Project, University of California, Davis. College of Agriculture and Environmental Sciences, Center for Design Research, Department of Environmental Design.

Medland, A.J. 1986. *The Computer-Based Design Process.* London: Kogan Page.

Miller, P.L. 1986. *Expert Critiquing Systems.* New York: Springer-Verlag.

Novitski, B.J. 1992. "Scanning Software." *Architecture.* (81) December. 95 - 97.

Novitski, B.J. 1992. "New frontiers in CAD." *Architecture.* January. 103 -105.

Novitski, B.J. 1993. "Software-assisted code compliance." *Architecture.* (82) February. 111-113.

Oxman, R.E. and R. M. Oxman. 1992. "Refinement and adaptation in design cognition." *Design Studies.* 13 (2) April. 117 - 134.

Parks Canada. 1983. *Researching Heritage Buildings.* Ottawa: Government of Canada.

Parks Canada. 1979. *The Evaluation of Historic Buildings.* H. Kalman (ed.) Ottawa: Government of Canada.

Prudon, T. 1992. "Saving houses of worship." *Architectural Record.* 179 (3) New York: McGraw-Hill Inc. 30 - 31.

Prudon, T. and S. Dalton. 1981. "The accessible path in historic properties." *Association of Preservation Technology Bulletin.* XIII (2). 31 - 39.

Rypkema, D. 1992. "Making renovation feasible." *Architectural Record.* 180 (1) New York: McGraw-hill. 26-27.

Saint-Aubin, J. 1990. "The image of built architecture." *Association for Preservation Technology Bulletin.* 22 (1/2). 43 - 53.

Seebhom, T. "Cad & the Baroque."*ACADIA'90 Proceedings.* P. Jordan (ed.) Hawaii: ACADIA. 79 - 97

Shaviv, E. 1986. "Generative and evaluative CAAD tools for spatial allocation problems." *The Computability of Design - 1986 Suny Buffalo Symposium on CAD.* A.C. Harfman, Y. Kalay, B.R. Majkowski, L.M. Swerdloff (eds.) Buffalo: State University of New York. Part 2.

Silverman, B.G. 1992. "Survey of expert critiquing systems: practical and theoretical frontiers." *Communications of the ACM.* 35 (4) April. 106 - 127.

Silverman, B.G. and T.M. Mezher. 1992. "Expert Critics in engineering Design: Lessions Learned and Research Needs." *AI Magazine.* Spring. 45 - 62.

Steinfeld, E. and Y. Kalay. 1990. "The impact of computer-aided desgin on representaiton in architecture." *Representation and Simulation in Architectural Research and Design.* 31 March - April 1. Robinson (ed.) State University of New York at Buffalo.

Stiny, G. 1990. "What is a design?" *Environment and Planning B: Planning & Design.* 17. Great Britain: Pion. 97 - 103

Sykes, M. and A. Falkner. 1971. *Canadian Inventory of Historic Buildings -- Training Manual.* 2nd ed. Ottawa: Department of Indian Affairs and North Development.

Vanier, D.J. 1992. "Details of a classification system to extract project-specific building codes." presented at *Joint CIB "Integrated Computer Aided Design" and*

"Computers and Building Standards" Workshops. 12-14 May 1992. Montreal, Canada.

Woodbury, R.F. 1986. "Strategies for interactive design systems." *The Computability of Design - 1986 Suny Buffalo Symposium on CAD.* A.C. Harfman, Y. Kalay, B.R. Majkowski, L.M. Swerdloff (eds.) Buffalo: State University of New York. Part 1.

Worling, J.L. 1992. "IRC resources -- electronic technical information." (Compact Disk) Ottawa: Institute for Research in Construction, National Research Council Canada.

GLOSSARY

The following are definitions of selected terms used in this thesis (acronyms are shown in square [ ] brackets).

A

American Standard Code for Information Interchange [ASCII]; a widely adopted standard (alphanumeric text) character encoding scheme, usually used to facilitate the exchange of digital data between different software and/or hardware environments.

analysis; the breakdown of a design problem into constituent elements or parts and explicit identification of the relationships between the elements. This process helps to break down larger problems into smaller, more manageable and understandable portions. It is used to clarify design problems, their organization, basis, and effects. As part of a design process this stage includes the explicit statement of design goals.

analyze design; a command in the prototype SGE system that is issued to expert system from within CAD environment to initiate a complete check of the CAD building design for building code compliance.

Architectural Attribute Coding [AAC]; a set of object attributes used to describe architectural properties or characteristics. This information is linked to objects in a CAD graphic, stored in a database and is used for design analysis by a knowledge base. The AAC includes physical, functional, signal, topological, and geometrical attributes. It is based on terminology used by domain experts to describe architectural entities.

Artificial Intelligence [AI]; a discipline within the field of computer science concerned with building computer programs that imitate human behaviour. More specifically, programs that perform tasks that require intelligence when performed by humans (e.g. game playing, inference, learning plan formation, speech recognition, natural language understanding). Expert systems are an application of AI.

as-built; actual physical characteristics of an existing building, as opposed to what may be described in design or construction documents and specifications (see also ***as-designed state***).

as-built files; "CAD files produced from the contract document files reflecting a building as constructed, including project change orders and site instructions. Some organizations refer to these files as ***record*** files" (PWC 1990, 1). In the context of redevelopment, measurements from existing heritage building would be used to update existing original building files (or to create new drawings when no drawings are available) at both the start (before construction), during and at the completion of the redevelopment process.

as-designed state; a data model describing an actual building design instance.

as-required (design) state; a data model of constraint data (such as a building code requirement).

associations (or relations, k-lines); a part of a semantic network diagram. Associations are labeled and directed lines representing knowledge linking data or concepts (nodes) together. Nodes are objects and their properties within a given domain of discourse. Associations include: class (Is A), sub-class (A Kind of), part-whole (Part Of), and property (Has).

attribute; a defined property of an entity or object (*Oxford* 1991, 26). "Text or numeric data field or variable attached to a symbol or geometric object on a CADD drawing. Examples of attributes are: model numbers for furniture symbols, door type codes for door symbols" (PWC 1990, 1).

attribute value; a numerical quantity or text that is assigned or determined by calculation or measurement for a given attribute. Usually includes type of data (format).

attribute values (sources of);

(1) ***explicit***; measured or intrinsic to geometry of an entity,
(2) ***calculated;*** by a procedure,
(3) ***assigned***; user-supplied (material, condition) and,
(4) ***default;*** in the absence of a value one is assigned automatically from preset values (best guess, common occurrence or average).

attribute values (types of);

(1) ***geometric:***
   (a) explicit (dimension, geometric character),
   (b) calculated (volume, area); and,
(2) ***non-geometric*** :
   (a) physical (material, color),
   (b) semantic (condition, quality).

automated design systems (generative design systems); computer programs capable of independently devising plans, patterns, or arrangements in order to achieve a functional purpose. Such systems are based on achieving a desired effect independent of a preset procedure(s).

B

bit mapped graphics; see *raster graphics.*

building product model; see *product data model.*

C

case-based reasoning; a problem-solving strategy where a unique problem is addressed using the same techniques used to successfully resolve previously encountered problems.

certainty factors; statistical probabilities of "correctness." Certainty factors are used where there are no absolutes such as "true" or "false" but rather concepts such as "mostly true," "sometimes false." In expert systems certainty is a mathematical property that the system maintains in association with the knowledge variables. Each time a conclusion is reached the composite value of certainties is calculated from all of the variables used to draw the conclusion. The system may conclude: "It is 75% certain that the building example given is Georgian in style" (see also *fuzzy logic).*

Computer Aided Design [CAD], Computer Aided Design and Drafting [CADD]; term used to describe computer programs that incorporate drawing tools to assist designers in the creation and visualization of architectural and engineering designs.

CAD file; "a group of drawing entities stored under a single name in a computer" (PWC 1990, 1).

class; "objects grouped together based on a common set of traits." (Carrico, et al. 1989, 307). A class contains templates or models of functional constraints, object characteristics, and/or contextual or object-to-object relationships that are common to groups of objects.

classification; process of grouping objects into classes.

classification paradigm; "The process of sifting out the correct or best alternative from among several choices. The goal of classification is to identify a general pattern or trend, not to delve into details" (Carrico, et al. 1989, 307).

classification (building codes); a categorization assigned to buildings based on occupancy, construction type, materials, height, siting and area used for determining the applicability of building code requirements.

client-server model; a programming technique that links together several programs through a single source application. Control of all other applications comes from within a single application.

closing error; a phenomena that occurs when start and end points do not match while creating a closed CAD entity based on predetermined but inaccurate measurements (usually taken on site from an original building).

concepts; an abstract or generic idea generalized from particular instances.

conceptual graph; see *semantic network.*

conceptual (product) models; domain knowledge and data represented as concepts linked together by associations. In the case of building design, conceptual models describe design elements and their attributes in terms of function or relationship to one another. Conceptual modeling is a fundamental approach for representing ideas about design knowledge at a non-specific level, independent of any particular system implementation technique. A concept described at a knowledge level generally has several representation options at the symbol level.

conflicts window; a rectangular area on a display screen which part of an image or file is displayed. Window are a means presenting the user with the results of different processes being done by the computer.

creation tools; commands available within a CAD environment to used generate graphic entities (such as lines, arcs, rectangles, and circles).

D

data sheet (or form); a display window in a computer program designed to resemble a form with blank spaces to fill in information. It is used to query the user for input data for a database. An example of a line on the form may be: "width of room: _____ mm." This is also known as *query by example* interface.

Database Management System [DBMS]; "This is a trade term referring to an application development environment that integrates database functions, report generators, user data entry interfaces, audit/recovery functions, and database administration duties." (Carrico, et al. 1989, 308).

decision trees; a tree where each branch or node represents a decision. A leaf is outcome of a sequence of decisions taken from root to leaf. Usually binary (two alternatives at each decision point such as "yes" or "no") (*Oxford* 1991, 123). See also *tree.*

design constraints; limits or restrictions that effect the manner in which a design is created - that which defines the satisfaction of the design goals.

design elements; components of a design.

Digital Heritage Building Record; a computer database of information relating to a historic structure including but not limited to: photographs (UGI), text (TTI), and measured drawings (SGI).

digital image; *see raster image.*

Digital Photogrammetry and Architectural Design [DIPAD] system; This is a system under development at the Swiss Federal Institute of Technology in Zurich, Switzerland. The system is used to acquire digital imagery at sufficient resolution using solid-state sensors to process the data using semi-automatic measuring techniques. The results of this processing creates a data structure useful for CAD files.

domain; The area of expertise of an expert. For example, the domain of an architect is building design.

domain of discourse; language and terminology used within a particular domain.

Dots Per Inch [dpi]; used to describe the resolution or fineness of detail in data that is recorded or output (as on a laser writer). It describes the number of points within an square inch area of the display or output for which the positions are accurately recorded/shown. The higher the dpi the greater the level of detail.

Drawing eXchange Format [DXF]; "a file format developed originally by the vendor of AutoCAD™. A DXF file is a neutral ASCII text file containing CAD drawing information that can be read by AutoCAD and many other CAD systems. Although binary versions of DXF files can be generated, these files are usually hardware and software dependent, and hence are not recommended for use in translation" (PWC 1990, 1).

E

embodied energy; energy attributed to an existing object that is the sum total of all the energy used to produce the materials used in it and processes used to assemble it.

entity (or ***object***); A geometric element or an item of data in a CAD file.

evaluation; (p. 24) Qualitative and quantitative judgment of the value of solutions relative to the satisfaction of criteria. The source of criteria may be determined by the designer (internal) or referenced from external sources. For example comparison of existing building design to building code requirements set by governing agencies (external) or against performance criteria set by the client (internal).

expert; "A person with extensive experiential and intuitive knowledge that is considered valuable" (Carrico, et al, 1989, 309).

expert system; a type of computer program designed to contain knowledge of an expert, usually narrow in scope and limited to a single specific subject domain (i.e. medical diagnosis).

explanation facility; part of a computer program used to provide help or additional information to the user. In some expert systems the rationale used by the computer to reach a conclusion is recorded so that the user can review it to see how the conclusion was reached.

F

field (database); "an item of data consisting f a number of characters, words, or codes that are treated together...a number of fields make up a *record*" (*Oxford* 1991, 173).

formal language; "A language with explicit and precise rules for its syntax and semantics. Examples include programming languages and also logics such as predicate calculus. Thus formal languages contrast with natural languages such as English whose rules, evolving as they have with use, fall short of being either a complete or precise definition of the syntax, much less the semantics, of the language" (*Oxford* 1991, 183).

forward-chaining; "A strategy inference engines may use forward chaining to manipulate information in a configuration problem Initial data is applied to the rulebase and rules that are eligible to be true are collected. Conflicts in recommendations made by eligible rules are resolved by changing data. The system then looks for new eligible rules. This process continues until all conflicts are satisfactorily resolved" (Carrico, et al. 1989, 309).

frames; "representation of an object as a data structure. Frames resemble tables in a relational database" (Carrico, et al. 1989, 310). "A knowledge representation formalism. A frame is a list of named slots. Each slot can hold a fact, a pointer to a slot in another frame, a rule for deriving the value of the slot or a procedure for calculating the value. Frames can be used to represent all the knowledge about a particular object or event. They are often arranged in hierarchies in which frames representing particular entities inherit their slot values from ancestor frames representing generic entities " (*Oxford* 1991, 187). Nodes in semantic networks (objects, properties, values) can be placed into frames. Frames transform nodes into DBMS-like records and allow them to include their own set of values in the form of attributes and attribute values (Carrara & Kalay, 1992). See also *object-oriented programming system* .

functional classification; where knowledge is classified entirely by what it does (function), rather than by the physical properties and characteristics of objects (structure) (Lucardie 1993, 107).

functional equivalence; two otherwise different objects or processes having the same purpose or result.

fuzzy logic; "a branch of logic designed specifically for representing knowledge and human reasoning in such a way that it is amenable to processing on the computer" (*Oxford* 1991, 192). Unlike standard logic theories, fuzzy logic allows for uncertainty, that is to say, more than just true or false, but includes concepts such as mostly true, not false, more or less true and so on (ibid.)

G

Geographic Information Systems [GIS]; interrelational (layered) database systems of graphic information.

goal-states; a description of desired relationships of design elements and attributes in relationship to one another.

graphic standards; universally adopted drafting conventions used by persons in architecture, engineering, building, and allied fields.

H

heuristic; self-taught through experimentation or observation. In the computer field this term refers to programs that employ self-learning (knowledge acquisition) to find the solution to a problem. Based on experience of working in a particular problem domain certain procedures are developed. Part of problem-solving involves the knowledge of what strategy or procedure is best suited to resolve a given problem (*Oxford* 1991, 206). In an expert system environment heuristic searches occur by the system searching through stored or generated plausible solutions and selecting the best ones using rules.

Historic Buildings Survey [HABS]/ Historic Engineering Record [HAER]; an ongoing program to secure and preserve records of historic American architecture and engineering/industrial processes. Mostly recorded using traditional methods, these extensive collections contain a wide variety of drawings, plans, photographs and other miscellaneous data relating to historic structures / industrial processes.

Historic Recording Management System [HRMS]; a database system used by Parks Canada to track inventory of historical building documentation.

hypermedia; a generic term used to describe techniques used to create and view information stored on a computer in a variety of formats (sound, graphics, video, text) in a non-sequential manner. In such environments the order in which information is viewed in controlled by the user. For example, a hypermedia record of a heritage building may allow the user to view pictures and drawings of the structure and listen to audio descriptions of the areas being viewed. From within this same environment the user may be able to access reference material explaining features of the style or period to which the building belongs.

I

inference engine; "The part of the knowledge program that scans, selects, and applies rules and data. A specific application occurs only when a rule and its related data pertain to a system goal or subgoal." (Carrico et al. 1989, 311) Whereas the knowledge contained in the knowledge base can be considered to be made of up facts and constraints, the inference engine acts as an interpreter when and how this knowledge is used to solve a particular problem.

inheritance; "An object or rule receives needed information from another object or rule in the knowledge system which already possess the needed information. The relationship between objects that can bequest/inherit information may be predefined or dynamic" (Carrico, et al. 1989, 311).

"In a hierarchy of objects an object generally has a parent object (superclass) at the next higher level in the hierarchy and one or more child objects (subclass) at the next lower level. Each object can have various attributes associated with it. The attributes can be local to that object or can be inherited from the parent object. Attributes can be further inherited by child objects (often without limit on the number of inheritances). In addition an object can be an instance of a ore general object (not a parent object) with which it shares variables and also inherits its attributes.

Inheritance is thus a means by which characteristics of objects can be replicated and instantiated in other objects. Inheritance is both static by abstract data type and dynamic by instantiation and value. Inheritance rules define what can be inherited and inheritance links define the parent an child of inheritance attributes" (*Oxford* 1991, 222).

Integrated Graphic Database [IGDB]; a database containing design information linked to objects in a CAD file.

Intelligent Computer Aided Design Systems [ICAD]; CAD programs integrated with Knowledge Base Systems, usually for the purpose of design analysis or generation.

interface; "The connection of a computer to a peripheral or outside device. Peripherals include other computers, printers, terminals and humans" (Carrico et al. 1989, 311).

Information Age; the current historical era which has been likened to the Industrial Revolution by authors such as (Kranzberg, 1989) in terms of the similar overwhelming impact the availability of inexpensive and readily available computer technologies has had on technical and social change.

K

knowledge; "Information or data organized into useful patterns or concepts" (Carrico, et al. 1989, 311). Expert knowledge is evidenced by a "...competence for selecting actions to realize goals. This competence is accomplished through knowledge of goals, knowledge of actions and knowledge relating goals to actions" (Lucardie 1993, 98). It is rational that if a particular action leads to a desired goal than that action will be selected (ibid.).

In (Carrara and Kalay, 1992) design knowledge is divided into three types: (1) descriptive (objects, functions, relations), (2) normative (goals, intents, constraints) and (3) operational (methods for selecting or giving values that get to goals). Design knowledge is therefore comprised of a set of attributes, beliefs, perceptions, relationships, aspirations, and methods that embody a designer's personal experiences, the professions shared experiences and the particular circumstances of a specific design project (ibid., 78).

knowledge acquisition facility; part of an expert system program that allows the system to acquire more expert knowledge about a domain (from a human expert or other source) and incorporate it into its knowledge base and/or inference engine.

Knowledge Assisted Design [KAD]; see *ICAD*.

Knowledge Base [KB]; "A collection of knowledge in a particular domain that has been formalized in the appropriate representation with which to perform reasoning. It is most frequently encountered within the context of expert systems, where the knowledge base might represent the rules and experience of an expert practitioner in that domain (e.g. medicine, or electronics). Typically knowledge will be expressed in production rule format and will represent the heuristic approach that a practitioner has developed through applying formal knowledge in the course of problem solving. Other knowledge representation formalisms are local formulae, semantic nets, and frames. Within expert systems there are two important classes of knowledge, static and dynamic. A static knowledge base consists of the domain knowledge necessary to perform problem solving and remains unchanged during the course of a problem-solving session. The dynamic knowledge base is used to store information relevant to solve a particular problem (e.g. the results of laboratory tests) and varies from one problem-solving session to the next" (*Oxford* 1991, 245).

Knowledge Base System [KBS]; "Computer systems which contain the understanding of information in addition to providing for the algorithmic transformation of data" (Carrico, et al. 1989, 311).

knowledge coding; "The process of converting knowledge representations into executable computer systems" (Carrico, et al. 1989, 312).

knowledge engineering; "The process of converting the results of knowledge acquisition into knowledge representations that can be coded" (Carrico, et al. 1989, 312).

knowledge level analysis; knowledge level analysis occurs before representation at the symbolic level (where one is concerned with encoding knowledge using data structures and processing techniques found in knowledge base or database technologies). During knowledge level analysis one is concerned only with what knowledge is present, generally as collections of facts serving to represent concepts. In the work of (Lucardie and Gelder, 1993) knowledge-level analysis is conducted according to the theory of *functional classifications*. Knowledge level analysis is often explored using *conceptual models*. The advantage to knowledge level analysis is that it provides a deeper understanding of domain knowledge and freedom from the limitations imposed by formal representations required to implement a system.

knowledge level approach; the examination domain knowledge based on semantic issues that can be used effectively for modeling, validation, and design while excluding concerns relating to the user-interface or implementation (Lucardie 1993, 107). See also *knowledge level analysis.*

knowledge level models; See *conceptual models.*

knowledge representation; "Any of the various formats used to structure human knowledge such as semantic networks, decision tables, etc." (Carrico, et al. 1989, 312). The representation of knowledge requires both symbolic representation of facts, concepts and beliefs and also the representation of the processes that acquire, interprets and apply them in appropriate circumstances (Carrara and Kalay 1992, 77).

L

language of design; an analogy for the formalized structuring of design elements and actions. The grammar or structure of a design language is based on design knowledge. "Sentences" are made up of component parts taken from a larger vocabulary of parts assembled in a particular manner according to rules of a design knowledge "grammar."

layers; "a characteristic or attribute of an entity on a CAD drawing used for classification and to control visibility. In some CAD systems the term *level* is used" (PWC 1990).

LEVEL5 Object; A commercially available expert system shell created by LEVEL5 Research, subsidiary of InfoBuilders.

LISt Processing [LISP]; "A programming language developed specifically for symbolic data manipulation. LISP originally provided the necessary computer functions to stimulate research in artificial intelligence. The use of LISP is an alternative to the use of shells for knowledge system programming" (Carrico, et al. 1989, 312).

M

meta knowledge; knowledge of how to use other knowledge.

method; see *procedure.*

microcomputer; "A computer device containing a central processing unit (CPU) constructed on a single microchip" (Carrico, et al. 1989, 312). Includes DOS/Windows-based Personal Computers [PC] and Apple's Macintosh.

MiniCAD+; Object-based CAD program by Graphsoft Inc.

N

natural language; computer terminology used to describe human language. In order for computers to understand human language they have to be able to process using knowledge of what is said (the language) and make a relationship to what it means.

neural network; "A form of computation inspired by the structure and function of the brain." One of the capabilities of neural network systems is to be "trained, using examples, to recognize certain patterns, for instance to classify objects by features" (*Oxford* 1991, 303).

Nijessen Information Analysis Method [NIAM]; a type of conceptual model used for diagramming architectural concepts and relationships.

O

object; "A collection of information in computer system design. An object may be a data structure, set of procedures, or combination of the two" (Carrico, et al. 1989, 314). Objects can be not only physical objects but phenomena or processes also.

object classes; see *class.*

object instance; a specific object.

object name; unique identification assigned to an entity.

object-oriented; based on discrete collections of information.

object-oriented design; "A software development technique in which the system is seen as a collection of *objects* that communicate with other objects by passing *messages*. Design is targeted toward defining the kinds of object, the methods (i.e. procedures of objects), and the message passed. OOD is based on the principle of information hiding" (*Oxford* 1991, 312).

object-oriented programming; "A programming methodology based primarily on data structures (cast as "objects") and secondarily on procedures that must exist to cause communication between data structures. Procedures are "bound" or owned by objects. Objects give rise to program modules" (Carrico, et al. 1989, 314).

object-oriented programming system; "A programming system that combines data abstraction, inheritance, and dynamic type binding. The central feature is the object, which comprises a data structure definition and its defined procedures in a single structure. objects are instances of a class, each instance having its own private instance variables. The class definition defines the properties of the objects in that class; hierarchical class structure are possible in which objects in a class inherit the properties of the parent class in addition to properties explicitly defined for the class. This facilitates sharing of code, since users can inherit objects from system collections of code.

The procedures of an object (often called methods) are activated by messages sent to the object by another object. Thus in an object-oriented programming system the basic control structure is message passing. The programmer identifies the real-world objects of the problem and the processing requirements of those object, encapsulating these in class definitions, and the communications between objects. The program is then essentially a simulation of the real world in which objects pass message to other objects to initiate actions" (*Oxford* 1991, 312).

objects (in object oriented computer architecture); "...data structures in memory that may be manipulated by the total system (hardware and software); they provide a high-level description that allows for a high-level user interface" (*Oxford* 1991, 311). Usually have names or labels that describe what type of object they are and what they can do.

objects (in object-oriented languages); "...an object is a package of information and a description of its manipulation, and a *message* is a specification of one of an object's manipulations.....a message merely specifies what a sender wants done, and the receiver determines exactly what will happen" (*Oxford* 1991, 312).

Optical Character Reader [OCR]; hardware equipment used to convert printed text into a digital form capable of being manipulated by word processor.

P

palettes; windows overlaid on the drawing environment. In the CAD program MiniCAD+ palettes are used by the user to input data or execute commands.

paradigm; "A representation for the internal model that humans use to think about something" (Carrico, et al. 1989, 314).

paradigm (for computers); "A model or example of the environment and methodology in which systems and software are developed and operated" (*Oxford* 1991, 327).

paradigm (general); " An outstandingly clear or typical example or archetype" (*Webster's* 1989, 853).

parameters (computers); "1. Information passed to a subroutine, procedure, or function. The definition of the procedure is written using *formal parameters* to denote data items that will be provided when the subroutine is called, and the call of the procedure includes corresponding *actual* parameters. 2. A quantity in a function or mathematical model whose value is selected or estimated according to the circumstances" (*Oxford* 1991, 329).

parameters (general); "any set of physical properties whose values determine the characteristics or behavior of something (i.e. parameters of the atmosphere such as temperature, pressure, and density)" or " something represented by a parameter: a characteristic element: broadly; characteristic, element, factor" or "limit, boundary" (as in the parameters of science fiction)" (*Webster's* 1989, 854).

parametric variation; ability to vary attributes describing the geometrical relationships of a CAD object.

physical elements; parts of a building design that are manifest or "touchable" such as the wall, floor, sink.

pixel; an abbreviated form for "picture elements." It is a single element in an array of many elements that contains information relating to a computer image. Each element contains specific information about a small region of the image such as brightness or color that is then displayed on a computer screen.

problem-structuring (or puzzle-making); Puzzle-making is a theory of problem-solving put forward by (Archea, 1986) that architects use to resolve design problems. Most problem-solvers start by stating explicit performance criteria against which they test alternate solutions. The architect, on the other hand, searches for a unique solution or effect that evolves through the design process. Therefore the architect looks for sets of combinatory rules (precedents) that will result in an internally consistent fit between a kit of parts and the effects that are achieved when those parts are assembled in a certain way (transformation). As a result of working this way researchers have found that architects have a unique ability over non-architects to solve ill-defined problems (Akin, 1988). Problem-structuring involves knowing how to break down an ill-defined problem into well-defined parts, resolving these parts, and the reassembling these partial solutions into a general solution for the entire problem (Simon 1969 in Aikin 1988).

procedure; a computer term referring to a section of a program that carries out a well-defined operation on data (also referred to as subroutine) (*Oxford* 1991, 358). "A set of steps to be followed to accomplish a task each time the task is to be done" (Carrico, et al. 1989, 314).

product (data) model; A universally adopted standard information model for specific (physical) products. Currently there is a large scale international effort by the International Standards Organization to set these standards. They are used for transfer of data in product design (in light of the development of Computer Automated Manufacturing). "The design decision process involves a large number of objects at different levels of abstraction. When viewed as a product model, the abstractions can be the products as a whole, as a composition of its functional systems or its subsystem, or as a collection of low-level individual parts of the design. Even a partial standardization of the underlying data structures would result in great savings in data communication and data exchange between applications and also ease human interaction with the data" (Bjork 1991) in (Gauchel, et al. 1993). The primary shortcoming is that product models lack a conceptual definition of design products. Additional shortcomings of product modeling is that it lacks clear levels, distinct goals, and mix contexts (i.e. what the use of the data model is for, such as CAD/CAM or design analysis by a KB). As a result product models can be so generic that they are not useful for specific applications (Lucardie and de Gelder, 1993). See also *conceptual product models*

production rule shell; "One expresses knowledge to a production rule shell in English like expressions (called heuristics or "rules of thumb"). The expert system conducts an investigation based on the content and interrelationship of the rules. The programmer of a production rule shell has analyzed and interpreted the knowledge of an expert in order to compose the set of rules which are collectively referred to as a *rule base* or *knowledge base*" (Carrico, et al. 1989, 314).

protocol (computers); "An agreement that governs the procedures used to exchange information between cooperating entities. In general, a protocol will govern the format of messages, the generation of checking information, and the flow control as well as actions to be taken in the event of errors" (Oxford 1991, 367).

protocol (general); "a code prescribing strict adherence to correct etiquette and precedence (as in diplomatic exchange and in the military services" or "a set of conventions governing the treatment of esp. the formatting of data in an electronic communications system" or "the plan of a scientific experiment or treatment" (*Webster's* 1989, 947).

Q

query-based (expert systems); describes a kind of interface used to collect information about a given problem by asking the user a series of questions. Responses determine the proceeding line of questioning is until the system can draw a conclusion.

R

raster (or bitmapped); A widely used technique for displaying graphics or pictures on the computer where images are built up from parallel lines consisting of smaller individual "dots" or elements called *pixels.*

raster grid; the location of each integer representing light intensity and/or color is organized on a square grid made up of smaller elements or *pixels.*

record (database); "A data structure in which there are a number of named components called fields, not necessarily of the same type. It my have variants in which some of the components, known as *variant fields* , are absent; the particular variant for a given value would be distinguished by a discriminant or tag field. The record is widely recognized as one of the fundamental ways of aggregating data (another being the array) and many programming languages offer direct support for data objects that take the form of records" (Oxford 1991, 382).

red-lining; an activity that takes place during the review of architectural and engineering drawings where errors are identified by marks with a red pencil.

redevelopment (historic structures); to integrate a new use into an existing structure without destroying historically significant qualities. This may require sensitive repair, maintenance, redesign or replacement of some elements of the building design.

relation, (or relationship, association); connection between two data elements. See *association.*

routine or parametric Design; A type of design process that involves the resolution of a well-defined design problem. It is characterized by "prototype refinement." In this process not only an example of design already exists but also a database of possible variations (Schmitt, 1990).

rubber-banding; drawing display technique incorporated within CAD systems that shows lines being drawn out or "stretched" from start point(s) designated by the user.

rule; "The expression of information in a cause-effect format, i.e., IF (premise) THEN (conclusion) ELSE (conclusion). Rules may contain precise knowledge, or they may be opinions or educated guesses. Rules have been used frequently through history to preserve unwritten knowledge and are often called "rule of thumb." Some times rules may use words or phrases other than IF and THEN. Proverbs are an example of rules in an alternative form. For example, consider the saying "The early bird catches the worm." This can be translated to IF you are alert and prompt, THEN you will have the best opportunity" (Carrico, et al. 1989, 316).

rule-based (expert systems); knowledge-based systems that employ rules as a method of representation of objects, their relations, properties, instances, and classes. Rules are used in conjunction with simple statements or facts, therefore, many rules are required to represent complex concepts. See also *production rule shell.*

S

Scaled Graphic Information [SGI]; drawn information where measurements are known or can be taken from (e.g. architectural plans, elevations, sections). Within the Smart Graphic Environment SGI is held in CAD files.

semantic description; There are a number of ways in which design concepts can be represented other than graphically. Such a description relies on an abstraction of information to describe the effect of the design (depending on what the information is being used for) as opposed to, for example, the actual measurements. In the context of this thesis this term is used to represent building information that is non-graphic based (such as the material, color, condition) properties (including inheritance) and relations of objects.

semantic network; "A means of representing relational knowledge as a labeled directed graph. Each vertex of the graph represents a concept and each label represents a relation between concepts. Access and updating procedures traverse and manipulate the graph. A semantic net is sometimes regarded as a graphical notation for logical formulae" (*Oxford* 1991, 408). "A data model consisting of objects, data, people, or any "things" (often called "nodes") that are interconnected by lines (links). The links must be labeled with their explicit semantic meaning such as "is a," "belongs to," is one of," etc." (Carrico, et al. 1989, 316).

The advantage of a higher level abstraction of design information is that it is capable of communicating and reason about design concepts in terms of objects and compositions that transcends the literal manifestations of any one design instance. Thus, it easier to compare designs and design constraints. This is one of a number of higher level abstractions of architectural form that are able to accommodate knowledge about the design without being overly specific.

semantics (computer); "That part of the definition of a language concerned with specifying the meaning or effect of a text that is constructed according to the syntax rules of the language" (*Oxford* 1991, 408).

semantics (general); the study of meanings (usually of or relating to meaning in language.

separation object; category of building objects that are "two-sided" or that have characteristics relevant to more than one object. A door connecting a hallway and a room is an example separation object.

shell (expert system); "a tool for building knowledge systems. Shells provided preconstructed essential utilities such as inference engines, user interfaces, and report generators allowing the developer to concentrate of the programming of the knowledge component" (Carrico, et al. 1989, 316).

show violations; command given to expert system through CAD environment that provides additional information about the problem and to selects graphic object in question.

slot; a data field in a frame (database)

Smart Graphic Environment [SGE]; a type of intelligent CAD system that provides designers with expert analysis of building design performance for the redevelopment of historical architecture. The components of the environment include: (1) an object-based CAD system, (2) integrated graphic database, (3) knowledge base system, and a (4) human architect (interface).

snap; drawing control feature of CAD programs that aligns graphic entities to grid or other designated points on the display screen.

space boundary object; category of building objects that connect objects that are spaces (hallways, rooms) with building elements (walls, floors, plumbing stack).

spatial systems; non-physical interior aspects of a building design (room, hallway, maneuvering space), organization of function or activity.

stripping; process by which an existing building is prepared for alterations, changes (for example rotted elements are removed, walls are repaired).

subclass (of objects); sharing the same attributes of a larger group as well as additional defining characteristics, a subset of the larger group. For example "window" is a subclass of the class "opening."

symbols; a collection of CAD graphic entities which are stored under a single name, and can be manipulated as a group and used on other drawings. They typically represent commonly used design elements that are placed in CAD drawings. Examples include a plan view of toilet or door swing. Symbols are stored in graphic database files called ***libraries*** that are included with CAD packages, created by the user themselves, and/or purchased separately from third party developers. Symbols often have attributes or ones can be attached to them. Other CAD systems may use the term ***block*** or ***cell*** instead of symbol.

synthesis; a putting together of elements and parts so as to form a whole involving the arrangement and combinations in a unique pattern or structure. In design this process involves the resolution of design problems through optimal functional layout of elements and the satisfaction of goals and intentions.

T

template matching; technique used to program computers to recognize objects in photographic images (raster files) by attempting to find similar patterns or instances of known objects by scanning the image for a match. Preset specifications identify pixel brightness and/or color and arrangement.

Text and Tabular Information [TTI]; written descriptions (non-graphic data) relevant to a building design (i.e. inspection reports, field notes, zoning designation, materials, condition).

topological analysis; describing a design form in terms of its component objects and relations. This type of analysis can be used to study placements or geometric configurations as point sets.

topology of design; a description of "design form in terms of objects and relations. The objects are the design elements; the relations, the connections between elements" (Coyne, et al. 1990, 123). For example: "connected to" or "above." These relationships can be depicted as a semantic network.

total station; a digital theolodite or survey instrument where measurements are recorded onto diskette.

tree; "A hierarchical structure of information consisting of nodes connected by branches. There is a unique node called the root that is not subsidiary to any other node" (Carrico, et al. 1989, 317).

triggering; an event that mandates the application of modern construction code requirements onto a rehabilitation project, such as a change in occupancy or use.

type descriptions; "Design knowledge, however, can also be depicted in terms of generic descriptions, as prototypes. We may therefore expect that syntactic knowledge can be represented in terms of type descriptions; there should be some body of knowledge by which this generic description can be instantiated -- translated into the description of a design instance. This knowledge could simply be embedded in the prototype description, in terms of acceptable ranges of possible values -- for example, that a small house has somewhere between one and three bedrooms. Such generic design descriptions are used informally in design manuals and guides" (Coyne, et al. 1990, 65).

type hierarchies; a diagram or graph showing the hierarchical classification of objects indicating the inheritance of attributes (parent to child).

U

Unscaled Graphic Information [UGI]; raster or bit mapped scans of images, illustrations, paintings, sketches, video and/or photographs that form part of a digital heritage building (visual) record.

user; The user of the computer system. In the context of this thesis the primary user of the Smart Graphic Environment would be an architect.

user (or human) interface; The interaction between human beings and computers. The design of human-computer interface impacts heavily on how easily a system is learned, performance speed in accomplishing tasks, error rates (and the ability to correct them independently), and user satisfaction. See also *interface.*

V

variable dependent; the value of one variable is dependent on the value or calculation using other variable(s), as in the case of the value of area being determined by the product of width and height.

variable independent; non-derived values.

verification (as paradigm); The process in which it is determined whether or not the objects given fulfill the requirements previously established that indicate a particular course or action. For example, in the case of a very historic window that is damaged the first priority is to select a suitable repair technique rather than wholesale replacement.

verisimilitude; the quality or state of having the appearance of truth (probable) or something depicting realism (*Webster's* 1989, 1310).

W

working memory; internal storage of a computer that can be directly addressed by operating instructions: main memory, cache.

window; "a rectangular area on a display screen in which part of an image or file is displayed." Windows are a means presenting the user with the results of different processes being done by the computer (Oxford 1991, 502).

REFERENCES

Akin, Ö. 1988. "Expertise of the architect." *Expert Systems for Engineering Design.* Academic Press, Inc.

Archea, J. (1986) "Puzzle-Making: What architects do when no one is looking." *Computational Foundations of Architecture.* Y. Kalay (ed.)

Bloom, B.S. 1956. *Taxonomy of Educational Objectives, Handbook I: The Cognitive Domain.* New York: Longmans, Green.

Carrara, G. and Y. Kalay 1992. "Multi-modal representation of design knowledge." *ACADIA Proceedings - Computer Supported Design in Architecture.* K. Kensek, D. Nobel (eds.). 77-80.

Carrico, M., J. Girard, J. Jones. 1989. *Building Knowledge Systems.* New York: McGraw-Hill Book Co..

Coyne, R.D., M.A. Rosenman, A.D. Radford, M. Balachandran, J.S. Gero. 1990. *Knowledge-based Design Systems.* Sydney: Addison -Wesley Publishing Co. Inc.

De Gelder, J.T.and G.L. Lucardie. 1993. "Knowledge and data modeling in CAD/CAM applications." *Design and Decision Support Systems in Architecture.* H. Timmermans (ed.) Dordrecht, the Netherlands: Kluwer Academic Publishers. 111-121.

Gauchel, J., L. Hovestadt , S. Van Wyk, R.R. Bhat. 1993. "Modular building models." *Design and Decision Support Systems in Architecture.* H. Timmermans (ed.) Dordrecht, the Netherlands: Kluwer Academic Publishers. 83 - 95.

Kranzberg, M. 1989. "IT as a revolution - the information age." *Computers in a Human Context.* T. Forester (ed.) Cambridge: MIT Press. 19-33.

Lucardie, G.L. 1993. "A functional approach to realizing decision support systems in technical regulation management for design and construction." *Design and Decision Support Systems in Architecture.* H. Timmermans (ed.) Dordrecht, The Netherlands: Kluwer Academic Publishers. 97 - 109.

Oxford Dictionary of Computing. (3rd ed.) 1991. Oxford: Oxford University Press.

Public Works Canada [PWC]. 1990. *CADD Layering for Architectural and Engineering Applications.* Ottawa: Public Works Canada, Architectural and Engineering Services.

Streilein A, H. Beyer, T. Kersten. 1992. "Digital photogrammetric techniques for architectural design." *International Archives of Photogrammetry and Remote Sensing.* Vol. XXXIX. Part B5. 825 - 831

Webster's Ninth New Collegiate Dictionary. 1989. Springfield, MA: Merriam-Webster Inc.