

3D BUILDING CADASTRAL DATABASE DESIGN
USING CityGML BUILDING MODULE:
A CASE STUDY IN THIMPHU CITY, BHUTAN



**A Thesis Submitted in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Geoinformatics**

Suranaree University of Technology

Academic Year 2015

การออกแบบฐานข้อมูลโครงอาคารแบบ 3 มิติ โดยอาศัยโมเดลอาคาร
แบบจำลอง CityGML กรณีศึกษาในเมืองทิมปู ประเทศภูฏาน



วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาภูมิสารสนเทศ
มหาวิทยาลัยเทคโนโลยีสุรนารี
ปีการศึกษา 2558

**3D BUILDING CADASTRAL DATABASE DESIGN USING
CityGML BUILDING MODULE: A CASE STUDY IN
THIMPHU CITY, BHUTAN**

Suranaree University of Technology has approved this thesis submitted in partial fulfillment of the requirements for a Master's Degree.

Thesis Examining Committee

(Asst. Prof. Dr. Songkot Dasananda)

Chairperson

(Assoc. Prof. Dr. Suwit Ongsomwang)

Member (Thesis Advisor)

(Asst. Prof. Dr. Sunya Sarapirome)

Member

(Dr. Panthip Piyatadsananon)

Member

(Prof. Dr. Sukit Limpijumnong)

Vice Rector for Academic Affairs
and Innovation

(Prof. Dr. Santi Maensiri)

Dean of Institute of Science

**3D BUILDING CADASTRAL DATABASE DESIGN USING
CityGML BUILDING MODULE: A CASE STUDY IN
THIMPHU CITY, BHUTAN**

Suranaree University of Technology has approved this thesis submitted in partial fulfillment of the requirements for a Master's Degree.

Thesis Examining Committee

S. Dasananda
(Asst. Prof. Dr. Songkot Dasananda)

Chairperson

Suwit Ong.
(Assoc. Prof. Dr. Suwit Ongsomwang)

Member (Thesis Advisor)

S. Sarapirome.
(Asst. Prof. Dr. Sunya Sarapirome)

Member

P. Piyatadsananon
(Dr. Panthip Piyatadsananon)

Member

Sukit Limpijumnong
(Prof. Dr. Sukit Limpijumnong)

Vice Rector for Academic Affairs
and Innovation

Santi Maensiri
(Prof. Dr. Santi Maensiri)

Dean of Institute of Science

ทาทิ : การออกแบบฐานข้อมูลอาคารแบบ 3 มิติ โดยอาศัยโมดูลอาคารแบบจำลอง CityGML กรณีศึกษาในเมืองทิมพู ประเทศภูฏาน (3D BUILDING CADASTRAL DATABASE DESIGN USING CityGML BUILDING MODULE: A CASE STUDY IN THIMPHU CITY, BHUTAN) อาจารย์ที่ปรึกษา : รองศาสตราจารย์ ดร. สุวิทย์ อ่องสมหวัง, 155 หน้า

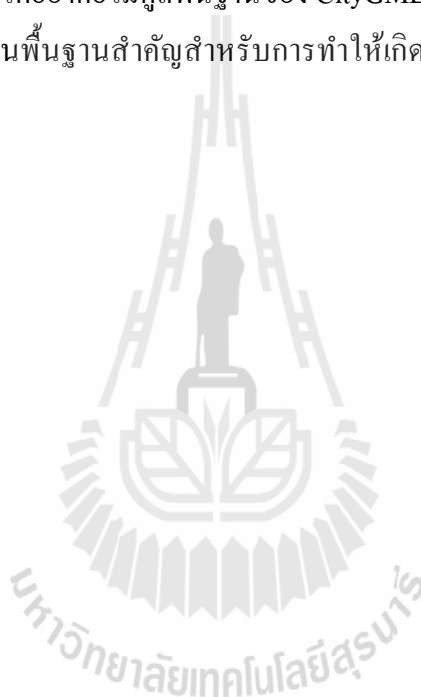
การขาดแคลนที่ดินว่างเปล่าและมูลค่าที่ดินที่สูงในเมืองทิมพูของประเทศภูฏาน ทำให้การใช้ประโยชน์ที่ดินเป็นไปอย่างเข้มข้น ประชากรเมืองนิยมเช่าซื้อห้องชุดมากกว่าการซื้อที่ดินว่างเปล่าเพื่อสร้างบ้านอยู่อาศัย จากสาเหตุดังกล่าวทำให้สิ่งปลูกสร้างทางกฎหมายต่างๆ ขยายตัวในแนวตั้ง และระบบการออกโฉนดอาคารแบบ 2 มิติในปัจจุบันไม่สามารถจัดการกับปัญหาดังกล่าวได้ ดังนั้น การออกแบบโฉนดอาคารแบบ 3 มิติ ในประเทศภูฏาน นับว่าเป็นเรื่องสำคัญ เพื่อตอบสนองการเปลี่ยนแปลงภาพเหตุการณ์ในการจัดการที่ดินที่กำลังเกิดขึ้นในปัจจุบัน รวมทั้งแนวโน้มที่เกิดขึ้น โดยทั่วไปในประเทศต่างๆ วัตถุประสงค์หลักของการศึกษาคือ เพื่อออกแบบแบบจำลอง UML ของ CityGML ADE สำหรับฐานข้อมูลการออกโฉนดอาคารแบบ 3 มิติ และเพื่อพัฒนารูปแบบการแลกเปลี่ยนข้อมูลและฐานข้อมูลการออกโฉนดอาคารแบบ 3 มิติ บนพื้นฐานของแบบจำลอง UML ของ CityGML ADE วิธีการศึกษาประกอบด้วย 4 องค์ประกอบคือ (1) การวิเคราะห์ (2) การออกแบบ (3) การทำให้เกิดผล และ (4) การทดสอบ

ผลการศึกษาลหลักประกอบด้วย การสร้างแบบจำลองอาคารแบบ 3 มิติ การสร้างแบบจำลอง UML ของ CityGML ADE สำหรับการออกโฉนดอาคารแบบ 3 มิติ การแลกเปลี่ยนข้อมูลและฐานข้อมูลอาคาร ในที่นี้ แบบจำลองอาคารแบบ 3 มิติ พร้อมด้วยผังโครงสร้างภายในได้ถูกสร้างขึ้นตามคลาสที่กำหนดด้วย CityGML โดยใช้ซอฟต์แวร์ SketchUp จากนั้น แบบจำลองอาคารแบบ 3 มิติที่ถูกสร้างขึ้นจะถูกแปลงออกเป็นคลาสต่างๆ ตามที่กำหนดในรูปแบบของไฟล์ gml ด้วยซอฟต์แวร์ Feature Manipulation Engine ในการศึกษาครั้งนี้ มีการเพิ่มคลาสรูปลักษณะพร้อมคุณสมบัติ ซึ่งประกอบด้วย คลาส “ApartmentUnit” “CommonPropertyUnit” และ “AccecesoryUnit” พร้อมกับหมายเลขอาคารและแปลงที่ดินเข้าในโมดูลอาคารของ CityGML จากนั้น ทำการสร้าง XML schema จากข้อมูลแบบจำลอง UML ของ Cadastre ADE ด้วยโปรแกรม ShapeChange อนึ่ง กระบวนการสร้าง XML schema และพจนานุกรมรายการรหัสอาศัยการประมวลผลด้วยกลุ่มคำสั่งที่สร้างขึ้นในโปรแกรม Enterprise Architect (EA)

ในขณะเดียวกัน การสร้างแบบจำลองฐานข้อมูลอาคารแบบ 3 มิติจะดำเนินการภายใต้โปรแกรม EA ซึ่งสนับสนุนฟังก์ชันการสร้างแบบจำลองโครงสร้างฐานข้อมูล ในที่นี้ ผังเค้าร่าง

UML ของตารางต่างๆ ถูกสร้างขึ้นแบบอัตโนมัติจากผังเค้าร่างแบบจำลอง UML แบบง่ายด้วย เครื่องมือการแปลงของ model driven architecture รายการรหัสที่ได้กำหนดไว้ล่วงหน้าจะถูกแปลง เข้าในตารางต่างๆ ที่ถูกกำหนดด้วยตัวระบุเฉพาะเพื่อเชื่อมโยงเข้ากับคอลัมน์ จากนั้น กลุ่มชุดคำสั่ง SQL DDL ที่สร้างขึ้นใน โปรแกรม EA จะถูกประมวลผลด้วย SQL Command Line เพื่อสร้าง ตาราง คอลัมน์ คีย์หลักและคีย์นอก และความสัมพันธ์ระหว่างตารางต่างๆ ใน Microsoft SQL Server 2008

จากผลการศึกษาสรุปได้ว่า สามารถออกแบบและพัฒนาต้นแบบฐานข้อมูลการออก โฉนด อาคารแบบ 3 มิติได้สำเร็จโดยอาศัยโมดูลพื้นฐานของ CityGML ภายใต้โปรแกรม EA ผลการศึกษา ที่ได้รับในครั้งนี้ นับว่าเป็นพื้นฐานสำคัญสำหรับการทำให้เกิดผลในการออกโฉนดอาคารแบบ 3 มิติ ในประเทศภูฏาน



TASHI : 3D BUILDING CADASTRAL DATABASE DESIGN USING
CityGML BUILDING MODULE : A CASE STUDY IN THIMPHU CITY,
BHUTAN. THESIS ADVISOR : ASSOC. PROF. SUWIT ONGSOMWANG,
Dr. rer. Nat. 155 PP.

3D BUILDING CADASTRE DATABASE DESIGN / CITYGML / ENTERPRISE
ARCHITECT / THIMPU CITY / BHUTAN

The scarcity of vacant land and high land value in Thimpu City of Bhutan has led to intensive use of space. People in the city find it more affordable to invest in the apartment rather than to buy a piece of land to build a house. This creates a different legal objects extending into vertical direction and the present 2D system cannot handle it properly. Therefore, the importance of implementation of 3D Cadastre in Bhutan is necessary to incorporate the present changing scenario in the land administration as well as future trend in development as prevailing in other countries. The main objectives of the study were to design UML model of CityGML ADE for 3D building cadastral database, and to develop a data exchange format and 3D building cadastral database based on the UML model of CityGML ADE. Research methodology consisted of four components: (1) analysis, (2) design, (3) implementation, and (4) testing.

Main results included construction of 3D building model, UML modeling of CityGML ADE for 3D cadastre, data exchange format, and building database. Herein 3D building model with interior layout was firstly generated according to the classes of CityGML under SketchUp software. The constructed 3D building model was

translated into their respective class using Feature Manipulation Engine as a gml file. In this study, building module of CityGML was extended with new properties included “ApartmentUnit”, “CommonPropertyUnit”, and “AccecesoryUnit” classes along with building number and its parcel number. After that, XML schema based on GML version 3.1.1 was generated from the UML model of Cadastre ADE using ShapeChange software. The whole process of XML schema and codelist dictionaries generation was activated through a batch file customized in Enterprise Architect (EA).

The modeling of the 3D building database was done in EA, which provided comprehensive functionality for modeling database structures. Herein the UML diagrams of tables were automatically generated from the simplified UML models using the tool model driven architecture transformation. The code lists defined were translated into tables assigned with a unique identifier to link to the column calling it. Finally, SQL DDL scripts generated in EA was executed using SQL Command Line to create tables, columns, primary and foreign keys, and relationship among the tables in Microsoft SQL Server 2008.

In conclusion, the prototype of 3D building cadastral database was here successfully designed and developed using CityGML as a base module in EA. This work provides a basis for 3D cadastre implementation in Bhutan.

School of Remote Sensing

Academic Year 2015

Student’s Signature _____

Advisor’s Signature _____

ACKNOWLEDGEMENTS

The presented research is part of my study for Master of Science in Geoinformatics work that has been carried out at Suranaree University of Technology, Nakhon Ratchasima, Thailand. The guidance and support of colleagues in the making of the thesis deserve a special mention. It is a pleasure to convey my gratitude to them all in my humble acknowledgement.

The first of all I am extremely indebted to my supervisor, Assoc. Prof. Dr. Suwit Ongsomwang. This work would not have been possible without his valuable guidance and support. His wisdom, knowledge, helpfulness and commitment for the work always inspired and motivated me throughout the study period. I gratefully acknowledge his support and advices throughout my stay in Thailand.

Thesis committee members, Asst. Prof. Dr. Songkot Dasananda, Asst. Prof. Dr. Sunya Sarapirome, and Dr. Pantip Piyatadsananon who supported me with their valuable comments and suggestions during my entire thesis work and despite their busy schedule, they always had a time to evaluate my thesis at different levels.

Most importantly, I would like to thank Royal Government of Bhutan for giving me the opportunity and Thailand International Cooperation Agency for awarding me a scholarship to pursue my master degree in Geo-informatics.

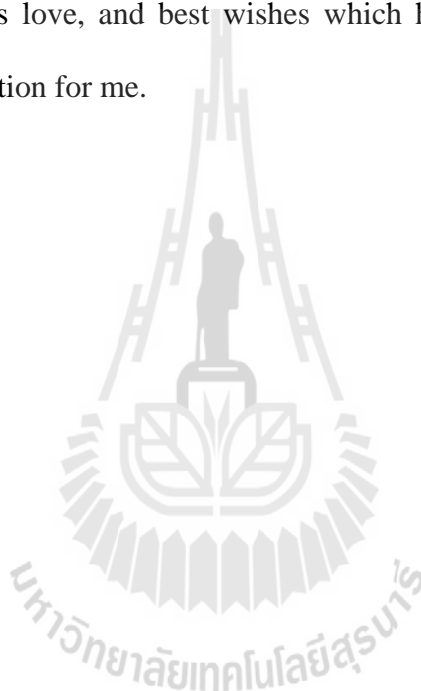
I would like to express my heartfelt gratitude to my colleague Mr. Tshering Wangchuck for helping me with the collection of data and Mr. Tandin Phuntsho for his valuable suggestion and sharing his know-how on the database management.

Besides this, I would like to thank Mr. Clemens Portele, Dr. Linda van den Brink, and support team of Safe software for their valuable contribution.

I extend my thanks to all the staffs and students of School of Remote Sensing for their help and support during my stay here. I also thank all colleagues who have knowingly and unknowingly helped me in the successful completion of this project.

Finally, I would like to express my deepest gratitude to my family for their moral support, endless love, and best wishes which have always been a source of inspiration and motivation for me.

Tashi



CONTENTS

	Page
ABSTRACT IN THAI.....	I
ABSTRACT IN ENGLISH	III
ACKNOWLEDGEMENTS.....	VI
CONTENTS.....	VII
LIST OF TABLES.....	XI
LIST OF FIGURES	XII
LIST OF ABBREVIATIONS.....	XVI
CHAPTER	
I INTRODUCTION.....	1
1.1 Background and significance of the study	1
1.2 Research objectives	4
1.3 Scope and limitations of the study	5
1.4 Study area.....	5
1.5 Benefit of the study	7
II BASIC CONCEPTS AND LITERATURE REVIEW.....	8
2.1 3D cadastre concepts	8
2.2 Basic concept of LADM.....	12
2.3 Overview of CityGML	15
2.4 Basics of UML	22

CONTENTS (Continued)

	Page
2.5 Literature review	24
III RESEARCH METHODOLOGY	35
3.1 Analysis component	36
3.1.1 Data collection	37
3.1.2 Research tools	37
3.1.3 Analysis of existing system	41
3.1.4 Requirement and specification of new 3D LIS	45
3.2 Design component	50
3.2.1 Construction of 3D data	50
3.2.2 UML modelling of CityGML ADE	51
3.3 Implementation component	54
3.3.1 Data exchange format	54
3.3.2 Specification of relational database	54
3.4 Testing component	55
IV RESULTS AND DISCUSSIONS	56
4.1 Specification of new 3D building database	56
4.1.1 New attribute definition	56
4.1.2 New class definition	57
4.2 Construction of 3D building model	58
4.2.1 Modeling of 3D building in SketchUp	58

CONTENTS (Continued)

	Page
4.2.2 Translation of SketchUp to CityGML in FME	60
4.3 UML model of CityGML ADE for 3D Cadastre	70
4.4 Data exchange format development	72
4.5 Building database	75
4.5.1 Simplification of CityGML and ADE for 3D Cadastre.....	76
4.5.2 Database modeling	80
4.5.2.1 Core model	81
4.5.2.2 Building model and ADE	84
4.5.2.3 Tables for geometry representation	88
4.5.2.3 Tables for codelist	90
4.5.3 Implementation of 3D building cadastral database	91
4.6 Testing	94
V CONCLUSIONS AND RECOMMENDATIONS.....	98
5.1 Conclusions	98
5.2 Recommendations	100
REFERENCES	102
APPENDICES	110
APPENDIX A XML SCHEMA OF CADASTRE ADE AND CODELISTS.....	111
APPENDIX B THE SQL SERVER 2008 DATA TYPES.....	119
APPENDIX C DDL SCRIPT FOR MS SQL SERVER 2008	126

CONTENTS (Continued)

	Page
CURRICULUM VITAE.....	155



LIST OF TABLES

Table	Page
2.1 LoD 0-4 of CityGML with their proposed accuracy requirements.....	19
3.1 Content of Cadastral data model of Bhutan.....	42
3.2 NLCS feature code for Building Model.....	43
4.1 List of new feature classes defined in CityGML	57
4.2 Data type mapping	76
4.3 Definition of Class names	83
4.4 Allowed integer values of objectClassID in the ThematicSurface table.....	86
4.5 Attributes determining aggregation types	89

LIST OF FIGURES

Figure	Page
1.1 Map depicting the study area: Thimphu city, Bhutan.....	7
2.1 Extend of ownership right.....	9
2.2 Building complex in The Hague, the Netherlands.....	10
2.3 Apartment complexes.....	11
2.4 An overview of Land Administration Domain Model.....	13
2.5 Content of Spatial Unit Package with associations to other basic classes.....	14
2.6 The CityGML modules.....	16
2.7 The five levels of detail (LOD) defined by CityGML.....	18
2.8 The two possibilities of modelling a building in LOD0 using horizontal 3D surfaces.....	20
2.9 Building model in LOD1 – LOD4.....	20
2.10 UML diagram of CityGML Building Model.....	21
2.11 UML overview.....	22
2.12 Aggregation and composition relationship in a class diagram.....	24
3.1 The overview of the research methodology.....	36
3.2 Content of Building Model of Bhutan.....	42
3.3 An example of title certificate issued: (a) Certificate, (b) Map, and (c) Legend..	44
3.4 Various cadastral objects related to strata titles in Malaysia.....	46
3.5 Example of drawing in strata title.....	47

LIST OF FIGURES (Continued)

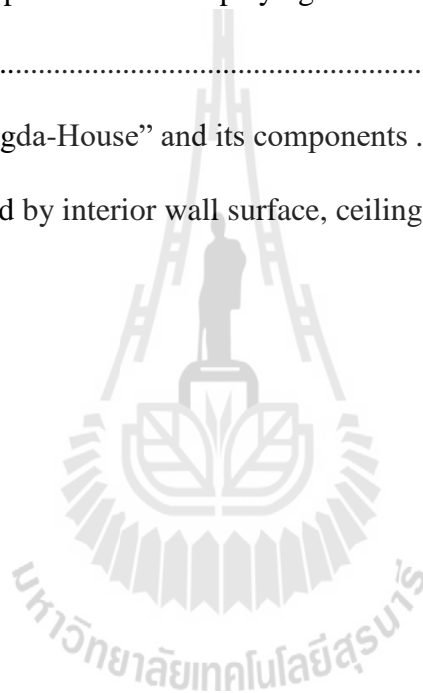
Figure	Page
3.6 Illustration of main components of cadastral building model	48
3.7 Illustration of the definition of interior face in example of Victoria, Australia (physical view) (left) and its representation in the subdivision plans (legal view) without depicting the structure.	49
3.8 Workflow for the construction of 3D data.	51
3.9 An overview of CityGML-IMGeo ADE.....	52
4.1 Example of the input data: (a) blueprint of building floor plan (CAD) and (b) the building footprint from cadastral data (.gdb)	59
4.2 Example of output of 3D data construction as 3D building model (.skp) under Trimble SketchUp.....	60
4.3 Overall SketchUp to CityGML translation diagram using FME	62
4.4 Workflow of SketchUp data preparation before writing into CityGML classes .63	63
4.5 Translation of LOD4 building.....	63
4.6 Translation of LOD1 building.....	64
4.7 Translation of room.....	64
4.8 Translation of ceiling surface.....	65
4.9 Translation of floor surface	65
4.10 Translation of interior wall surface	66
4.11 Translation of exterior wall surface	66
4.12 Translation of doors	66

LIST OF FIGURES (Continued)

Figure	Page
4.13 Translation of windows.....	67
4.14 FME translation log file	68
4.15 Output of the FME translation in FME Data Inspector.....	69
4.16 UML diagram of Cadastre ADE	71
4.17 Process of ShapeChange	72
4.18 Configuration file of ShapeChange in XML format	74
4.19 The log file displaying the results and messages from ShapeChange tool	75
4.20 UML diagram of core model.....	77
4.21 UML diagram of building model	78
4.22 The simplified UML diagram of Core model	80
4.23 The simplified UML diagram of building model and ADE	80
4.24 Tables of database schema of core model.....	82
4.25 CityObject envelope storage as a 3D rectangle specified by (left: the two black points with minimum and maximum coordinate values respectively) and (right: black polygon using five points).....	84
4.26 Building database schema	85
4.27 Geometry hierarchy.....	89
4.28 Tables of codelist	91
4.29 List of tables under BuildingDB database in MS SQL Server.....	92
4.30 Diagram of BuildingDB database (tables, column, keys, and relationship) ...	93

LIST OF FIGURES (Continued)

Figure	Page
4.31 XML schema validation result	94
4.32 Top view and [inset] list of buildings	95
4.33 A clip from output GML file displaying list of hierarchical ordering of building components.....	96
4.34 Building “Wangda-House” and its components	97
4.35 Rooms bounded by interior wall surface, ceiling surface, and floor surface..	97



LIST OF ABBREVIATIONS

2D/3D	=	Two Dimensional/ Three Dimensional
3DCDM	=	3D Cadastral Data Model
3DCityDB	=	3D City Database
ADE	=	Application Domain Extension
BIM	=	Building Information Models Planning
B-REP	=	Boundary Representation
CAD	=	Computer Aided Drawing
CityGML	=	City Geography Markup Language
DBMS	=	Database Management System
DDL	=	Data Definition Code
EA	=	Enterprise Architect
FIG	=	International Federation of Surveyors
FIG	=	International Federation of Surveyors
FK	=	Foreign Key
FME	=	Feature Manipulation Engine
GIS	=	Geographic Information System
GML	=	Geography Markup Language
ID	=	Identifier
IITB	=	Indian Institute of Technology Bombay
IMGeo	=	Information Model Geography

LIST OF ABBREVIATIONS (Continued)

ISO	=	International Organization for Standardization
JRE	=	Java Runtime Environment
KML	=	Keyhole Markup Language
LADM	=	Land Administrative Domain Model
LIS	=	Land Information System
LOD	=	Level of Detail
MDA	=	Model Driven Architecture
MSL	=	Mean Sea Level
NLCS	=	National Land Commission Secretariat
OCL	=	Object Constraint Language
OGC	=	Open Geospatial Consortium
OOXML	=	Office Open XML
PIN	=	Plot Identifier Number
PK	=	Primary Key
RRR	=	Right, Restriction, and Responsibility
SDI	=	Spatial Data Infrastructure
SKOS	=	Simple Knowledge Organization System
SQL	=	Structured Query Language
UML	=	Unified Modeling Language
VP-UML	=	Visual Paradigm for UML
WYF	=	World Youth Foundation

LIST OF ABBREVIATIONS (Continued)

XMI	=	XML Interchange Files
XML	=	Extensible Markup Language
XSD	=	XML Schema Document



CHAPTER I

INTRODUCTION

1.1 Background and significance of the study

Urbanization has driven the population growth centralizing in the metropolitan area (Chiang, 2012). On the other hand scarcity of vacant land in urban areas has led to the intensive use of the space. This intensive utilization of space has led to high-rise buildings and complex shaped structures as well as underground structures. These constructions extend the use of land from surface level to three dimensional levels, and challenge the traditional land registration and cadastral systems with multi-ownerships of properties which are located on the same piece of land (Mingru, 2007). Although property has been located on top of each other for many years, it is only recently that the question has been raised as to whether cadastral registration should be extended into the third dimension (van Oosterom, Stoter and Lemmen, 2005). Many countries such as the Netherlands, Israel, Greece, Norway and Sweden already have recognized this problem, and fulfilled the studies on the 3D cadastre as a way to solve these problems. As a solution, a 3D cadastre system can represent rights in three-dimensional space (Park, Lee, and Li, 2009).

Many efforts have been made to provide quality in the implementation of land administration all over the world. Many jurisdictions, organizations and software developers have developed their own cadastral data model. Examples of data

modelling developments are the core cadastral data model, FGDC Cadastral Data Content Standard for the National Spatial Data Infrastructure, ArcGIS Parcel Data Model, ePlan, Legal Property Object, and ISO 19152- Land Administrative Domain Model (LADM) (Aien, Kalantari, Rajabifard, Williamson, and Shojaei, 2012). Among the list, LADM has been widely used and many countries like Japan, the Netherlands, Portugal, Indonesia, Korea, Cyprus and other have adopted the LADM to develop their country profile of land information system (LIS).

The LADM is a model developed in the field of land administration as an international standard by ISO/TC 211. The LADM provides an abstract conceptual schema with three basic packages: parties (such as people or organization), administrative rights, restrictions and responsibilities (such as property rights) and spatial units (such as parcels, building and networks), with the latter having one sub-package: surveying and spatial representation (Elia, Zevenbergen, Lemmen, and van Oosterom, 2011). LADM supports both the 2D and 3D legal property, but it does not define the storage of physical property. Instead the physical property is stored in some external class, which is outside the scope of LADM.

On the other hand, there exists an Open Geospatial Consortium standard called CityGML (City Geography Markup Language), an application independent information model and exchange format for objects of 3D city landscape. It maintains semantic, geometry, topology and the appearance of objects (Stadler and Kolbe, 2007). Although CityGML is an international standard model and it supports five level of detail for spatial data of urban areas, it is limited to represent and manage the legal information needed in cadastre. However, the Application Domain Extension (ADE) method of CityGML makes it possible to extend the rich thematic and

geometry quality of CityGML to hold the legal information regarding the 3D cadastre. Accordingly Gozdz, Pachelski, van Oosterom, and Coors (2014) have studied the integration of LADM and CityGML to build 3D cadastral system and concluded that it is possible to introduce the semantic representation of land administration from the LADM into the CityGML.

Land recording system in Bhutan date back to 17th century, developed for the purpose of tax assessment. The most commonly known land record was prepared during the reign of the first King of Bhutan in year 1919 (NLC, 2014). Over decades, the land registration system was reviewed repeatedly and it has evolved to accommodate the changing scenario in land administration as well as the advancement in the field of technology. Presently National Land Commission Secretariat (NLCS) have completely updated and fully operated cadastral system, but only in 2D.

In recent years, the urban areas of Bhutan have witness inclined growth in population and infrastructures. These have ultimately led to extensive use of space of urban areas which are generally stretched along the narrow valleys of Bhutan. Moreover due to the high land value and low income of the people, people find it more affordable to invest in the apartment rather than to buy land. The business of real estate has boomed and people have started buying apartment. Such situation has raised issues regarding the provision of legal assurance on the utilization of space (Hendriatiningsih, Abdulharis, and Hernandi, 2012). The people who invested in apartment wishes to register their properties, but the current land administration system cannot support the situation of strata registration. Apart from apartment, there are other forms of 3D situation like underground parking, flyover, utility network

(such as power, telecommunication, water, and sewerage) growing in the cities without a proper system to map it. Bhutan's growing economy mainly depends on the sale of clean electricity generated through the growing industry of hydropower. These hydropower industries built their powerhouses inside mountain creating another form of 3D situation.

Therefore, the importance of implementation of 3D Cadastre in Bhutan is necessary to incorporate the present changing scenario in the land administration as well as future trend in development as prevailing in other countries. After analyzing the characteristics and reviewing papers, it can be concluded that the integration of LADM and CityGML can handle the existing 2D information as well as the growing 3D situation in Bhutan. The implementation of some of the classes from the LADM with the classes of CityGML will be studied here to fulfill the need of growing changes in land administration in Bhutan. Thereby, this study will design a CityGML ADE for 3D building Cadastre in Bhutan and its realization into the existing relational database under the realm of NLCS.

1.2 Research objectives

The primary objective of the research is to study the development of 3D building cadastral database in Bhutan using LADM and CityGML models. The specific objectives of the research are as follows:

- (1) To design UML model of CityGML ADE for 3D building cadastral database, and
- (2) To develop a data exchange format and 3D building cadastral database based on the UML model of CityGML ADE.

1.3 Scope and limitation of the study

The focus of this research is to design a 3D building cadastral database for LIS of Bhutan. Based on the objectives, the scope and limitations of the study are as follows:

(1) Considering the demand and the urgent requirement of 3D cadastre of NLCS, only urban area was considered. Later it will be extended phase-wise to the rest of the country.

(2) Although it is necessary to study all the thirteen thematic modules of city landscape presented in CityGML, detailed description on each model is beyond the scope of this research. Here only the building module is elaborated and adopted to fulfill the need of 3D situation in Thimphu City.

The design of 3D cadastral building database is based on the architectural style and 3D cadastral situation in Bhutan. Even the code list is based on national requirement. Therefore, the limitation of the study is that its implementation might not work in other countries. Another limitation or more precisely the difficulty of the study is acquiring the existing floor plan. All the building drawings submitted by the individual land owner for the construction approval are housed in the municipal office, but they do not have the copyright to share them. Available dataset collected through personal contact are here used in this study.

1.4 Study area

Thimphu city, Bhutan was chosen as the study area for implementation of 3D building cadastral database based on LADM and CityGML Models (Figure 1.1). Thimphu city is the capital and the largest urban center in Bhutan. It spans within an

area of 26 sq. km, stretching along the Wang River. It is located at 27°30' N latitude and 89°30'E in central western part of Bhutan.

It is the most pullulated and fast growing city in Bhutan. Rapid expansion following the pattern of rural exodus has resulted in considerable rebuilding in the city center and mushrooming suburban development elsewhere. Presently the whole of Thimphu is estimated to house around 6,500 buildings (kuenselonline, 2015) and population of about 104,000 (thimphucity, 2015). It consists of varieties of infrastructure development with different form of land administration. Recently the real estate business has boomed in Thimphu due to the scarcity of land and high land value. People started investing in apartment, which creates a different situation for the land administrator to secure the right of ownership to individual. Unfortunately the present LIS does not support the registration of apartment, though the lands are registered as shared property among the owners.

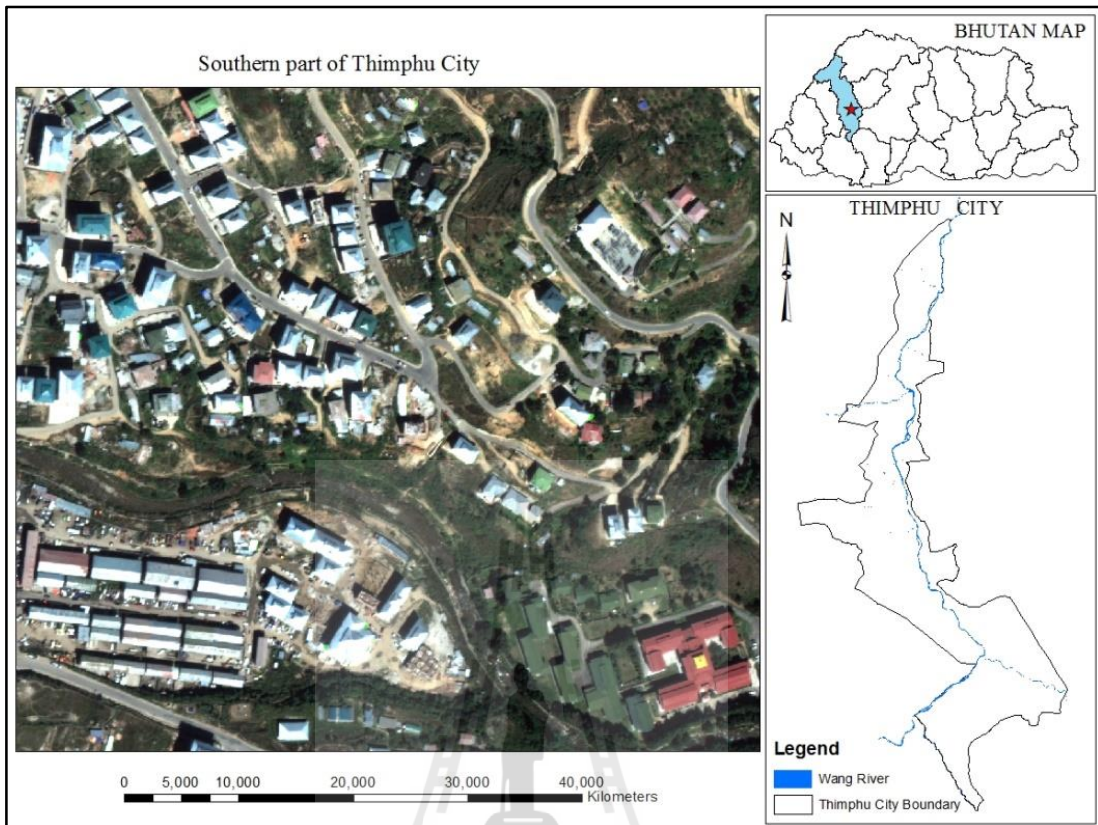


Figure 1.1 Map depicting the study area: Thimphu city, Bhutan.

1.5 Benefits of the study

(1) Insight information on different methods and models available in the field of 3D cadastre.

(2) A designed prototype of 3D building database system for NLCS of Bhutan, for future implementation/use.

CHAPTER II

BASIC CONCEPTS AND LITERATURE REVIEW

This chapter explains the concepts and theories, covering the four main topics related to the research: (1) 3D cadastre concept, (2) basic concept of LADM, (3) overview of CityGML, and (4) basic of UML.

2.1 3D cadastre concepts

This section introduces some concept of 3D cadastre from the thesis work “3D Cadastre” by Stoter (2004). Stoter (2004) defines 3D cadastre as a cadastre which registers and gives insight into rights and restrictions not only on parcels but on 3D property units. A 3D cadastre should be capable of storing, manipulating, querying, analysis, updating, and supporting the visualization of 3D land rights, restrictions and responsibilities (Aien et al., 2012). According to Aien, Kalantari, Rajabifard, Williamson, and Wallace (2013), 3D cadastre:

- (1) represent the spatial extent of ownership boundaries in the third dimension of height where layered and stratified ownerships exist,
- (2) facilitate registration of 3D property rights,
- (3) support land development processes including issuing of permit plans in dense urban areas especially for large scale developments such as bridges and tunnels which cross above or under other developments,

(4) provide reliable information for decision makers, and

(5) utilize as a basic layer to integrate with other information layers such as 3D city models (CityGML), Building Information Models (BIM), transportation and utility networks, land use controls, and delivery of services for different applications.

A 3D property unit is that bounded amount of space to which a person is entitled by means of real rights. It can be a land parcel, a building, an underground structure, or a superficies. Though 3D legal property is bounded by 3D space in some countries, but most of the countries have no limitation on the extension, that is the ownership can reach as far as the owner's interest. As shown in the Figure 2.1 the right of ownership can extend from the center of the Earth to the infinity sky above.

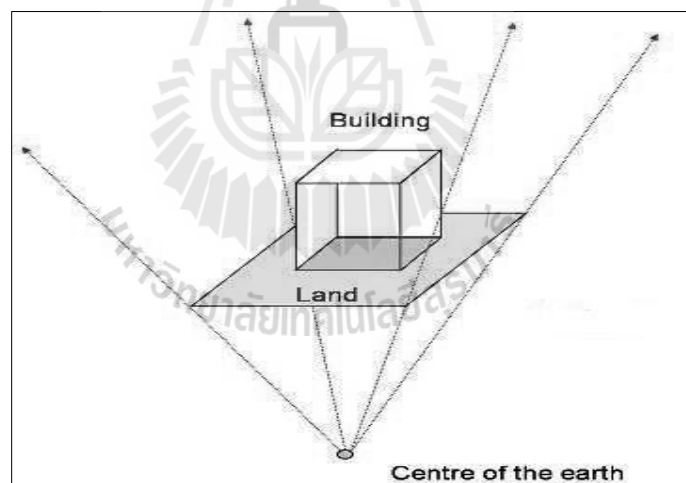


Figure 2.1 Extend of ownership right (<http://resource.unisa.edu.au/mod/book/print.php?id=55928>).

Generally, 3D cadastral system involves related subjects like 3D registration, 3D modelling, 3D geodatabase, and 3D visualization. 3D registration deals with maintaining both the spatial and non-spatial information of 3D objects. The types of rights associated with a 3D component being the main non-spatial information can be

broadly differentiated as: (1) right of ownership; (2) limited ownership rights such as right of superficies, right of long lease, right of easement; (3) right to an apartment or condominium right; and (4) joint ownership.

Figures 2.2 and 2.3 illustrate examples of some common 3D situations and the rights associated with it, as observed in the Netherlands. Figure 2.2 shows an example of 3D situation of a building over a highway in the city of The Hague. In this example there exists only one building with one owner, but it requires three parcels to establish the legal status of the whole building. “Ing Vastgoed Belegging BV” possessed an unrestricted right of ownership on parcel 1718, a right of superficies on parcel 1719 and a right of long lease on parcel 1720. The municipality holds a restricted right of ownership on parcels 1719 and 1720.

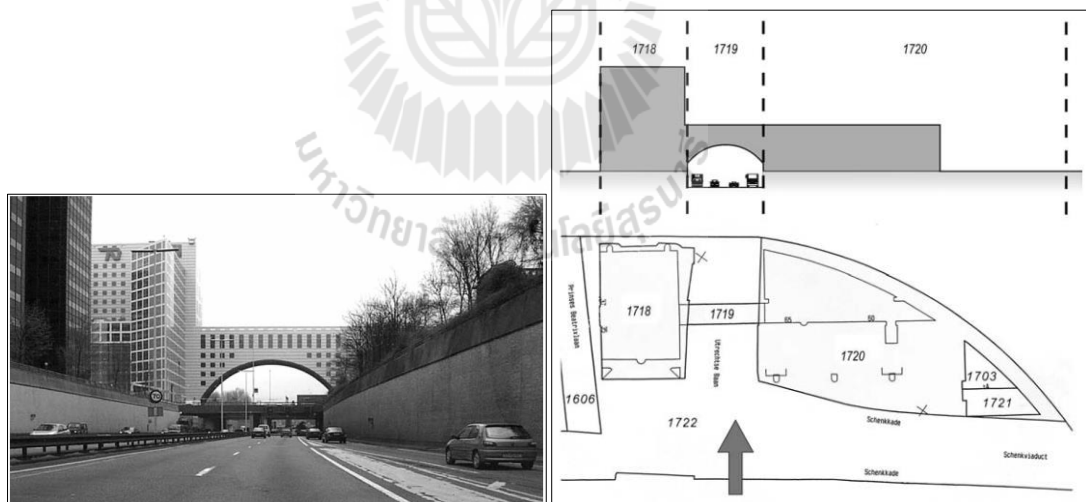


Figure 2.2 Building complex in The Hague, the Netherlands (Stoter, 2004).

Figure 2.3 shows an example of apartment ownership. As shown in the right picture, the whole building is divided into three apartments. One apartment is located

on the ground floor, and the two other apartments are located on the second and third floor, next to each other, with an entrance on ground level.



Figure 2.3 Apartment complexes (Stoter, 2004).

3D cadastre can be interpreted in many ways ranging from a full 3D cadastre supporting volume parcels, to the cadastre in which limited information is maintained on 3D situations (Stoter and van Oosterom, 2006). Basically it can be distinguished into three fundamental categories:

(1) Full 3D cadastre – This defines the property rights in 3D space. The 3D space is subdivided into volume parcels partitioning the 3D space. The legal basis, real estate transaction protocols, and the cadastral registration should support the establishment and conveyance of such 3D rights.

(2) Hybrid cadastre – This mean preservation of the 2D cadastre and the integration of the registration of the situation in 3D and being part of the 2D cadastral data set. This results in a hybrid solution with the legal registration of 2D parcels and a registration of the 3D situation.

(3) 3D tags linked to parcels in current 2D cadastral registrations – Here the 2D cadastre holds reference to a digital or analog representation of 3D situation. The difference with the hybrid cadastre is that the 3D representations are maintained separately, not integrated with the cadastral geographical data set.

2.2 Basic concept of LADM

The contents of this section are based on the final draft version ‘ISO/FDIS 19152: Geographic information – Land Administration Domain Model (LADM)’ released in 2012.

The LADM is a conceptual schema covering basic information-related components of land administration (including those over water and land, and elements above and below the surface of the earth). LADM enables the combining of land administration information from different sources (i.e. organizations) in a coherent manner (Lemmen, van Oosterom, Uitermark, Zevenbergen, and Cooper, 2011). The standard provides an abstract model with three packages and one sub-package covering all basic aspect of land administration system. These packages are: Party Package (green), Administrative Package (yellow), and Spatial Unit Package (blue) (Figure 2.4). The Surveying and Representation Sub-package (purple and red) is a sub-package of the Spatial Unit Package. These classes support 2D and 3D spatial units as well as RRR, which are applicable to facilitate 3D property objects registration and visualization for land and space registration (Budisusanto, Aditya, and Muryamto, 2013).

For the purpose of this research, the focus is the Spatial Unit package. The Spatial Unit package contains the classes concerning spatial elements such as land parcel, buildings, utilities networks and attributes describing them (area, volume, geometry) (Bydlosz, 2013). The main classes in this package are SpatialUnit, LegalSpaceNetwork, and LegalSpace-BuildingUnit (Figure 2.5).

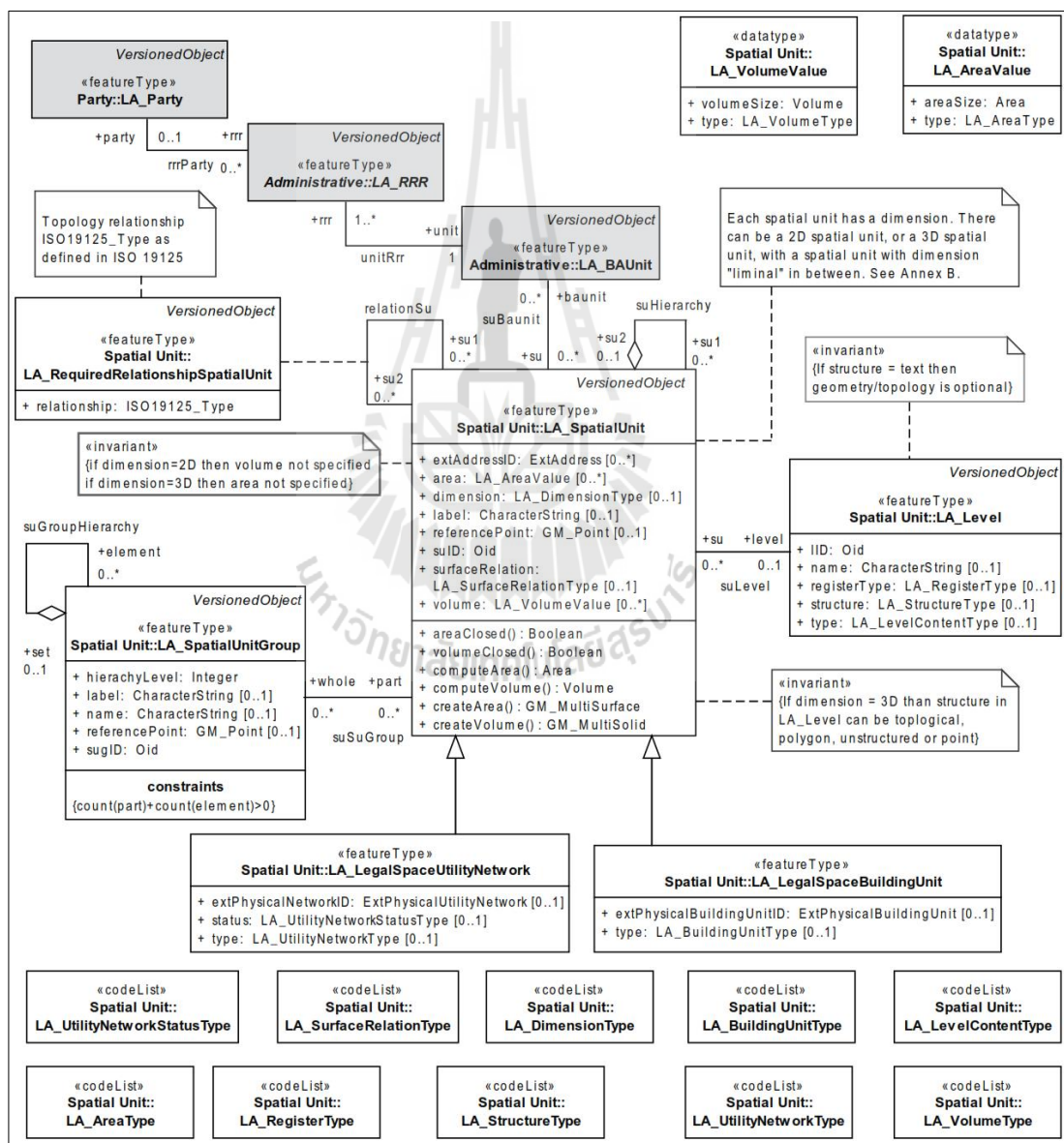


Figure 2.5 Content of Spatial Unit Package with associations to other basic classes (ISO/TC211, 2012).

A ‘spatial unit’ can be represented as a text (“from this tree to that river”), a sketch, a point (or multi-point), a line (or multi-line), a polygon (or multi-polygon) representing a single area (or multiple areas) of land (or water) or, more specifically, a single volume of space (or multiple volumes of space) (Lemmen and van Oosterom, 2013). Spatial unit class is structured in such a way to support the creation and management of basic administrative units. It can be further specialized into LegalSpaceBuildingUnit or LegalSpaceUtilityNetwork. These sub-classes are useful to describe legal spaces within building/utilities, which coincide with the physical space of the building (or parts of it)/utilities (Ronsdorf, Wilson and Stoter, 2014).

2.3 Overview of CityGML

The content of the section is based on an adopted international standard of the Open Geospatial Consortium (OGC) ‘The CityGML Encoding Standard document version 2.0.0’ published on April 04, 2012.

CityGML is an open data model and XML-based format for the storage and exchange of virtual 3D city objects. It is an application schema for the Geography Markup Language version 3.1.1 (GML3), the extendible international standard for spatial data exchange issued by the OGC and the ISO TC211. The aim of the development of CityGML is to reach a common definition of the basic entities, attributes, and relations of a 3D city model. CityGML provides a standard model and mechanism for describing 3D objects with respect to their geometry, topology, semantics and appearance, and defines five different levels of detail (LOD). CityGML enables lossless information exchange between different systems and users (Snowflake Software, 2015). Its rich application includes urban and landscape

planning, 3D cadastre, environmental simulation, mobile telecommunications, disaster management, homeland security, and many more.

The CityGML data model consists of a core module and thirteen thematic extension modules. The core module defines the basic components of the CityGML data model. Primarily, this comprises abstract base classes from which all thematic classes are derived. It also includes non-abstract content common to more than one extension module, for example basic data types. The extension modules cover specific thematic fields of the virtual 3D city model including: Appearance, Bridge, Building, CityFurniture, CityObjectGroup, Generics, LandUse, Relief, Transportation, Tunnel, Vegetation, WaterBody, and TexturedSurface (Figure 2.6).

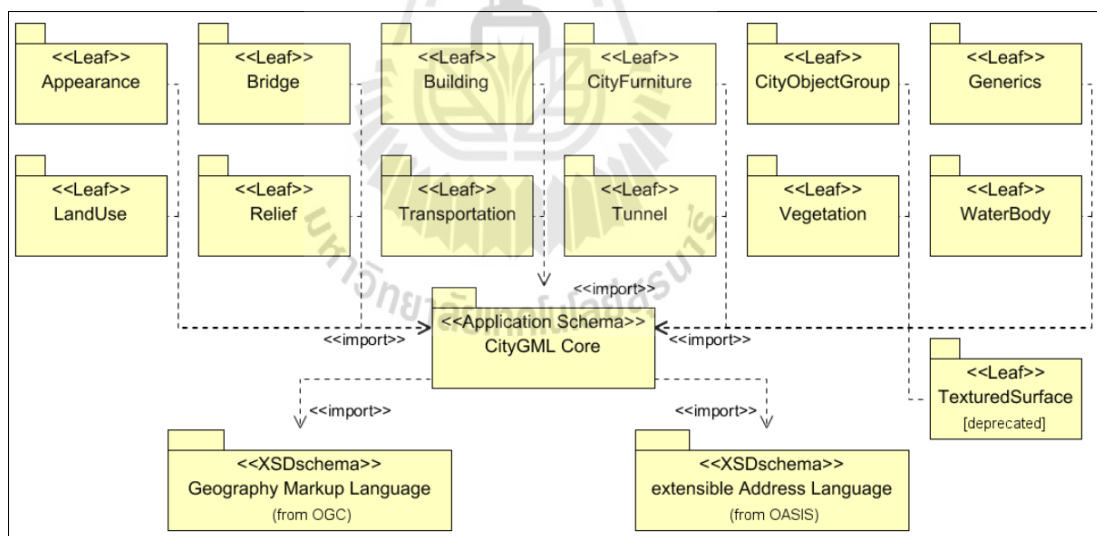


Figure 2.6 The CityGML modules (OGC, 2012).

Most of the name of thematic modules clearly specifies its contents, except modules such as Appearance, Generics, and CityObjectGroup. Appearances represent the visual data as well as themes such as infrared radiation, noise pollution, or earthquake-induced structural stress. Generics module represent the 3D objects, which

are not covered by the thematic classes of CityGML. The current version of CityGML does not include, for example, explicit thematic models for embankments, excavations and city walls. The CityObjectGroup module provides a grouping concept for CityGML. According to user-defined criteria, arbitrary city objects can be aggregated in groups to represent and transfer as part of a city model.

Another option to Generic module, for extensions of the CityGML data model for specific fields, can be realized using the Application Domain Extensions (ADE). ADE specifies addition to the CityGML data model through the introduction of new feature types, attributes, geometries, and associations. Also new elements can be added to the existing feature types with the ADE. Dsilva (2009) specified five advantages of ADE over generic objects and attributes:

- (1) ADEs can be formally specified,
- (2) Extended CityGML instance documents can be validated against the CityGML and its ADE schema,
- (3) More than one ADE can be used in the same dataset,
- (4) ADEs may be defined for one or more CityGML modules which provides a high flexibility, and
- (5) ADEs can be defined and standardized by communities that are interested in specific application domains.

CityGML also provides code lists to allow some predefined values for the CityGML attributes. This code list is allowed for extension and even replacement, to suit the requirement or preference of a national code list. However, CityGML do not provide a normative way to structure code lists. There exist some prominent choices such as GML dictionary and Simple Knowledge Organization System (SKOS).

CityGML differentiates five consecutive Levels of Detail (LOD), where objects become more detailed with increasing LOD regarding both their geometry and thematic differentiation (Figure 2.7). LODs are characterized by differing accuracies and minimum dimensions of objects (Table 2.1). CityGML files can - but do not have to - contain multiple representations (and geometries) for each object in different LOD simultaneously. Furthermore, objects can have external references to corresponding objects in external datasets such as telecommunication, census, and many more.

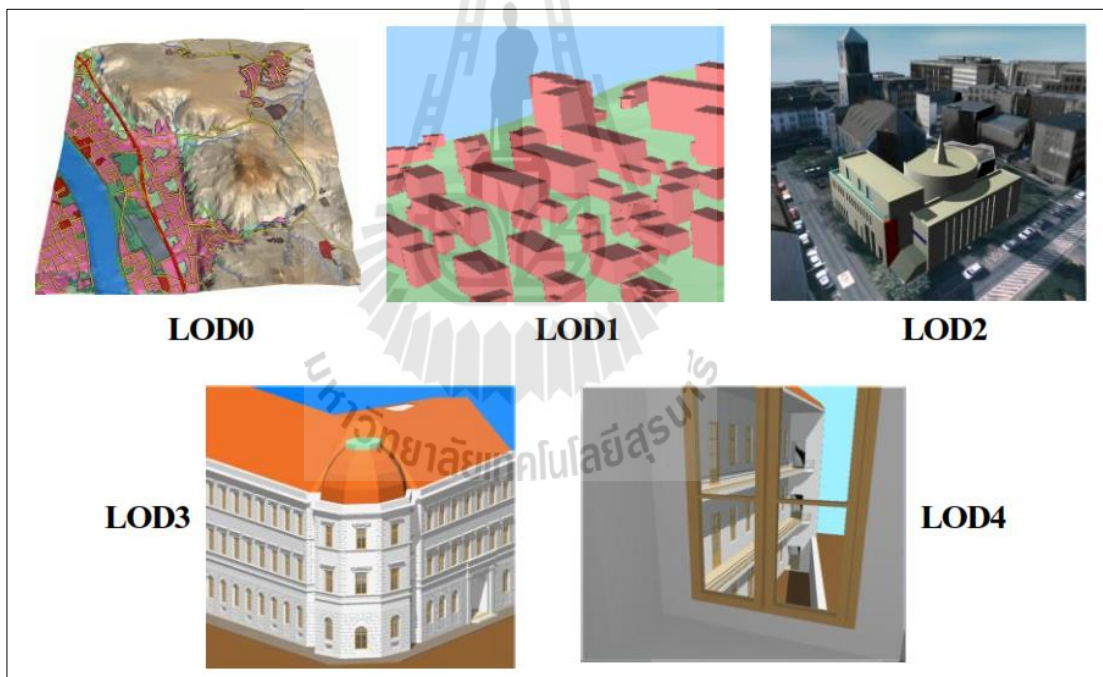


Figure 2.7 The five levels of detail (LOD) defined by CityGML (OGC, 2012).

Table 2.1 LOD 0-4 of CityGML with their proposed accuracy requirements (Snowflake Software, 2015).

	LOD0	LOD1	LOD2	LOD3	LOD4
Model scale description	regional, landscape	city, region	city, city districts, projects	city districts, architectural models (exterior), landmark	architectural models (interior), landmark
Class of accuracy	Lowest	Low	middle	high	very high
Absolute 3D point accuracy (position / height)	lower than LOD1	5/5m	2/2m	0.5/0.5m	0.2/0.2m
Generalization	maximal generalization	object blocks as generalized features; >6*6m/3m	object blocks as generalized features; >4*4m/2m	object as real features: >2*2m/1m	constructive elements and openings are represented
Building installation	No	No	yes	represented exterior features	real object form
Roof structure/representation	Yes	Flat	differentiated rood structures	real object form	real object form
Roof overhanging parts	Yes	No	yes, if known	yes	yes
CityFurniture	No	important objects	prototypes, generalized objects	real object form	real object form

The Building Module which is related to the study is one of the most detailed thematic concepts of CityGML. It allows for the representation of thematic and spatial aspects of buildings and building parts in five levels of detail. Buildings may be represented in LOD0 by footprint or roof edge polygons (Figure 2.8). LOD1 is the well-known blocks model comprising prismatic buildings with flat roof structures. In contrast, a building in LOD2 has differentiated roof structures and thematically differentiated boundary surfaces. LOD3 denotes architectural models with detailed wall and roof structures potentially including doors and windows. LOD4 completes a LOD3 model by adding interior structures for buildings (Figure 2.9).

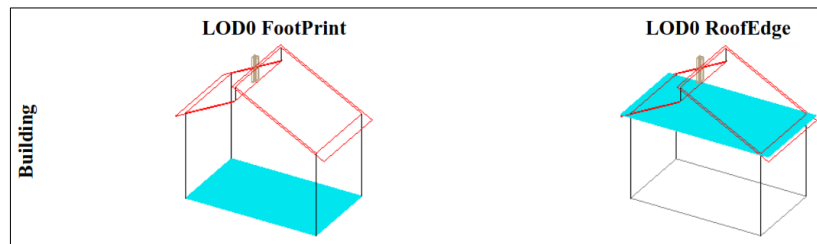


Figure 2.8 The two possibilities of modelling a building in LOD0 using horizontal 3D surfaces (OGC, 2012).

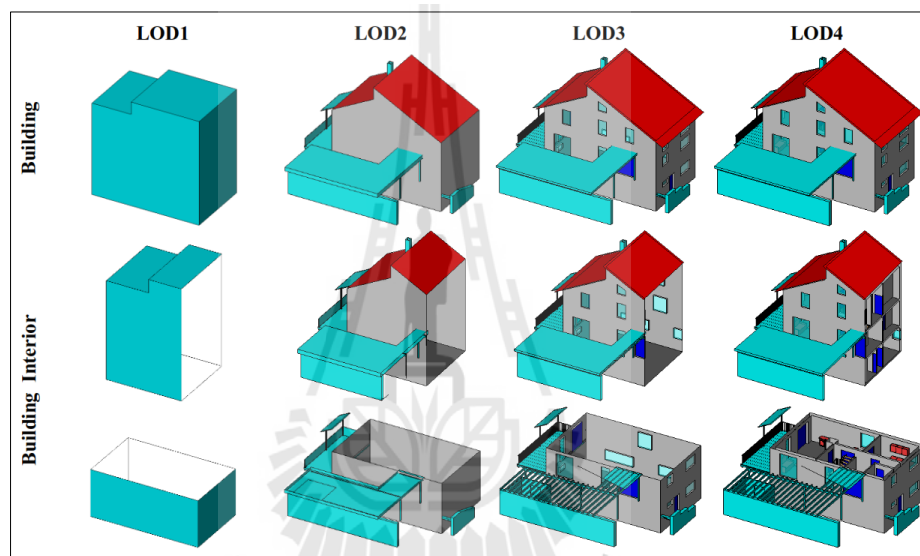


Figure 2.9 Building model in LOD1 – LOD4 (OGC, 2012).

The pivotal class of the Building Model (Figure 2.10) is AbstractBuilding, which is a subclass of the thematic class Site and transitively of the root class CityObject. The class AbstractBuilding has two specializations namely Building and BuildingPart. As shown in Figure 2.10, both classes inherit the following attributes: the class of the building, the function, the usage, the year of construction, the year of demolition, the roof type, the measured height, and the number and individuals heights of the storeys above and below ground. A building may have zero or more building installation objects such as chimneys, stairs, antennas, or balconies, which

2.4 Basics of UML

LADM uses Unified Modeling Language (UML) class diagram to design its conceptual model. OGC have also adopted UML as a modeling language to model the extension of CityGML. In regard to that, basic skill of UML class diagram is required to understand the concept of LADM and CityGML standards.

The UML is a general-purpose modeling language in the field of software engineering, which is designed to specify, visualize, and document models of software systems as a standard way. It was adopted as ISO standard in year 2000. Figure 2.11 shows an overview of the UML class diagram. It consists of classes (super class and sub-class), attributes, operations, constraints, associations (inheritance/ specialization), and code list.

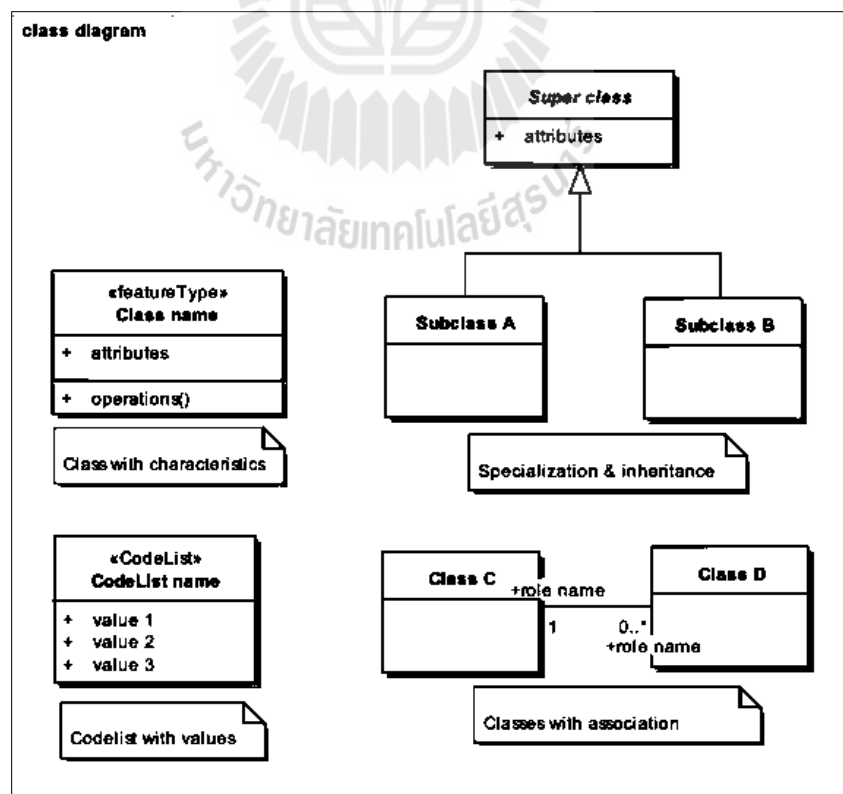


Figure 2.11 UML overview (van den Brink, Stoter, and Zlatanova, 2013).

The class diagram shows a collection of classes, interfaces, associations, collaborations and constraints. A class is an element that defines the attributes and behaviors that an object is able to generate. Classes are typically modeled as rectangles with three sections: the top section for the name of the class, the middle section for the attributes of the class, and the bottom section for the methods of the class (Agilemodeling, 2015).

Generalizations, aggregations, and associations are the relationship defined between classes to reflect inheritance, composition or usage, and connections respectively. Association is the general relationship type between elements. The connector may include named roles at each end, cardinality, direction and constraints. A generalization is used to indicate inheritance between parent class and child classes. Aggregation is a form of association used to depict a whole/part relationship, whereas composite is a stronger form of aggregation. If the parent of a composite aggregation is deleted, usually all of its parts are deleted with it; however a part can be individually removed from a composition without having to delete the entire composition. Aggregation relationships are shown by a white diamond-shaped arrowhead pointing towards the target or parent class and composition by a black diamond-shaped arrowhead (Figure 2.12).

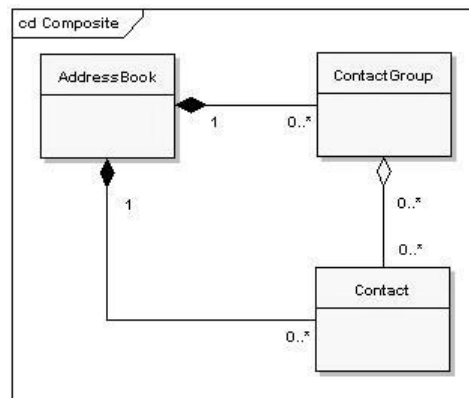


Figure 2.12 Aggregation and composition relationship in a class diagram (http://www.sparxsystems.com/resources/uml2_tutorial/uml2_classdiagram.html).

2.5 Literature review

Zulkifli, Rahman, and van Oosterom (2014) discussed 3D objects registration and modelling for cadastral objects within the Land Administration Domain Model (LADM) framework in the paper “3D strata objects registration for Malaysia within the LADM framework”. The paper first defined the various types of strata objects in Malaysia supported by the Strata Title Act and Strata Management Act. The strata objects included (1) building and building parts (all in 3D within a single lot), (2) land parcel (with house no more than 4 storeys within a single lot), which can be refined with (3) parcel unit, (4) accessory unit, and (limited) (5) common property unit including support for provisional and multilayer/underground aspects. Based on the LADM, the researchers proposed and developed a conceptual model as well as the associated technical model for Malaysia country profile. The country profile was explained in two parts: spatial part and administrative (legal) part. The spatial part of Malaysian country profile was developed for the 3D spatial units represented by building, utility and lot. All the classes were defined as a specialization of LADM

class “LA_SpatialUnit”. The legal part contained Party and Administrative package of LADM. The technical model was realized in Oracle and uploaded with study area data located at World Youth Foundation (WDF) building in the state of Melaka and some land parcels around the building. After uploading the sample data, the prototype frontend developed on Bentley Microstation was used to visualize the geometry and query using SQL. This paper serves as an example to the other countries that have the strata title system.

Gozdz, Pachelski, van Oosterom, and Coors (2014) conducted a study on the application of CityGML in the representation of geometric and descriptive data about buildings for the support of cadastral and administrative tasks. OGC (2012) stated that the CityGML, which provides a geographical information model for urban landscapes, not only represents the shape and graphical appearance of the 3D city objects, but also addresses the representation of the semantic and thematic properties. For that reason, the standard can be applied for the 3D representation of buildings. However, the CityGML does not contain any features describing the legal information about spatial objects needed for cadastral purposes. In support, El-Mrkawy and et Östman (2012) also stated that the CityGML standard with its current status does not have capabilities to build 3D cadastral system. Therefore, this paper presented the proposal for the integration of the LADM and CityGML at the conceptual level to support both the legal and physical description of a 3D building model needed in a cadastral system. Application Domain Extension provides an extension capability to incorporate the requirement of specific application in the CityGML data model. The method used for development of the CityGML-IMGeo ADE as Dutch 3D national standard (accepted as OGC Best Practice) was adopted here for preparation of

CityGML-LADM ADE. To link the legal space occupied by buildings and their physical counterparts, two classes representing buildings: PL_LegalSpaceBuilding, and PL_Building were introduced. Conceptually, class PL_LegalSpaceBuilding representing the legal space was defined as subclasses of _CityObject, whereas classes PL_Building and PL_BuildingPart, representing the physical space were defined as subclasses of their counterparts from the CityGML. Information about entities and property rights was represented as separate classes as LA_Party, LA_RRR, and LA_BAUnit in CityGML-LADM ADE. Through this study they have confirmed that the CityGML provides a flexible conceptual model, which can be adapted to land administration domain, particularly for supporting the spatial concepts required for cadastral system.

van Oosterom (2014) reported the results of the second phase of the 3D Cadastre and LADM investigations in context of possible future renewal of the Cadastral database at the Survey of Israel. The first phase of the investigations was reported on 25 March 2014 and it covered two studies: (1) an overview of the international state of the art of three-dimensional cadastre and (2) current cadastral procedures, land model and database. The second phase continued with the recommendations from the first phase of the investigations and addresses a series of 3D Cadastre/LADM topics such as standards, procedures, case studies, SDI, LADM country profile, data transfer, DBMS schema, query and visualization. ISO/TC211 standard LADM was used to define the conceptual model of the Israel 3D country profile, supporting both cadastral map and land registry (legal) information. During the development, various meeting was held and communicated through emails, to discuss the main scoping questions raised by the International Federation of

Surveyors (FIG) Working group 3D Cadastral. (1) The first question on types of 3D cadastral objects was answered by deciding to support both related to constructions (buildings, pipelines, tunnels, etc.) and any part of 3D space (airspace, subsurface). (2) 2D floor plan was used for the representation of apartments/condominium buildings as well as related facilities such as storage or parking instead of 3D parcel, which will be implemented in the future. (3) 3D parcels for infrastructure objects such as long tunnels, pipelines, and cables was decided to represent in block-wise and joined during the registration. (4) For representation of 3D parcel, legal space was decided to have its own geometry. Considering the discussed scope and various design considerations, the first step taken was mapping of the key concepts in LADM and their counterparts in the actual registrations. The conceptual design was then converted to technical design, which can be used for implementation in the context of a database schema and/or exchange format. A seven steps workflow was identified as follows: (1) Survey or mutation plan and (2) B-rep provides the spatial data sources for the new 3D parcels; (3) data transfer standard was considered; (4) automated quality check to assess the completeness of 3D data as well as its geometrical and topological errors; (5) technical database model was realized using an Oracle database and ESRI ArcGIS; (6) for data visualization and (7) dissemination, standards like 3D pdf, X3D was considered.

In the 5th Land Administration Domain Model Workshop held in Kuala Lumpur, Malaysia in 2013, Andrade, Carneiro, and Santos (2013) presented the full specification of LADM for relational database for the Republic of Cape Verde. The objective of the research was to propose a conceptual and abstract model based on the LADM and to test its application in a specific database model. Initially, an abstract

model “LADM_CV” profile was proposed considering the national reality and the decree that establishes the judicial regime of the building cadastre in Cape Verde. The specification of the conceptual and abstract model through a relational database was carried using the platform VP-UML, synchronize the UML profile to a relational entity model and finally exported to the Postgres. The VP-UML was chosen due its capacity to execute the whole modeling process, from the construction of the UML diagrams until its implementation in a database. Despite that fact this platform showed an absence of OCL language, which is used as one way of validating the model. An empty database project was created (LADM_CV) in the PostgreSQL 9.1 with the spatial extension Postgis 1.5. The data base server was configured, and the user name and password for the super user were defined. Immediately after, connection between the Visual Paradigm for UML 10.1 and the PostgreSQL was setup using the name and password of the superuser through the server created previously in the PostgreSQL. After the connection, the model in UML was transformed into a relational database schema (Entity-Relation Model). The SQL code was generated from the schema in VP-UML, which is subdivided into two groups: DDL (Data Definition Language) and DML (Data Manipulation Language). The first was responsible for the creation, modification and management of tables, while the second creates the structures which will contain the data, that is, the feeding and management of data.

van den Brink, Stoter, and Zlatanova (2013) in their paper “UML-based approach to developing a CityGML Application Domain Extension” tried to find out the best approach for modelling CityGML-IMGeo ADE, the national 3D standard of the Netherlands. The researchers evaluated six alternatives based on their pros and

cons. Alternative 3 stated as “Add properties in a subclass in the ADE package but suppress this subclass from the generated XML Schema” stand out to be best approach for modelling the CityGML ADE and later adopted by OGC as the best practice (OGC, 2014). They described the reasons for their choice as (1) IMGeo being an extension of CityGML, therefore adding the IMGeo classes as subclasses of CityGML and also extra attributes are appropriate; (2) usage of sub-classes are understandable to people; (3) the approach confirmed to relevant rules of UML, the ISO 19100 series and OGC; and (4) the approach is more in the line with the current geo-information modelling approach in the Netherlands. The procedures followed for modelling IMGeo as an ADE of CityGML in UML are followed. Firstly, UML model of CityGML is recreated in the modeling tool Enterprise Architect based on OGC (2012). Secondly, a conceptual mapping is done between the CityGML and IMGeo at the semantic levels to identify the equivalent classes of IMGeo in the CityGML. Two followed solutions to map the classes are (1) remodel the IMGeo as much as possible to find an equivalent CityGML class, and (2) classes, which cannot be remodeled, are extended as subclass of one of the CityGML classes with stereotype <<featureType>>. All IMGeo classes modeled as subclasses of CityGML classes are assigned stereotype <<ADEElement>> to suppress from appearing in the XML Schema. Thirdly, definition of code lists based on the notational list is done in Simple Knowledge Organization System (SKOS). The code lists are maintained in the UML model and XML structured code lists are generated using ShapeChange tool. Fourthly, representation of geometry and topology of features are decided. All LODs are defined with same accuracy alike in CityGML. The 2D representations of the buildings are used to represent the building geometries on the surface and the

geometries above or below are modeled separately. For topology a general rule defined in IMGeo standard, which stated that the 2D objects at the surface level must form a topological structure of the Netherlands without gaps or overlaps, are copied in CityGML-IMGEO ADE. Finally, a XML Schema is generated from the UML design using Java tool ShapeChange.

Aien, Kalantari, Rajabifard, Williamson, and Wallace (2013) proposed a 3D cadastral data model (3DCDM) in their paper 'Towards integration of 3D legal and physical objects in cadastral data model'. The research started by investigating the existing cadastral data models and concentrated on the problems: its inefficiency to facilitate effective representation and analysis of 3D data, negligence of interrelation of 3D legal objects with their physical counterparts, and lack of semantic properties. The researchers analyzed all the existing cadastral data models supporting 3D properties such as ArcGIS Parcel Data Model, ePlan, the South Korean cadastral data model and LADM. They expressed that only LADM had solution to represent 3D properties, but LADM did not employ proper 3D geometry primitives such as solid geometry, which facilitates 3D representation, volumetric calculation, and 3D spatial analysis. The paper also stated that the current cadastral models do not support semantics, which facilitate interoperability and integration of 3D legal objects with their physical counterparts. To overcome the above mentioned problems regard to the current cadastral data model, they proposed a new data model called 3DCDM, which composes of two main components namely LegalPropertyObject and LegalPhysicalObject. Legal Property Object was proposed by Kalantari (2008) where the Right and Parcel classes were combined into one class to facilitate incorporation of wide range of legal entities into the cadastral system. Here, physical property

objects such as building, tunnel, land, and utility network was linked to their legal object by a unique identifier. The relevant module building was explained further breaking it down into its classes such as building, building parts, and structural component to describe the physical as well as the legal extend. Geometric elements, solid and surface, were used to represent the physical objects. Through the development of the 3DCDM model they aimed to achieve a conceptual framework for 3D cadastres, represent key components and their relationships, facilitate subdivision of buildings and strata developments, and integrate physical counterparts of the legal objects to support a multipurpose 3D cadastre.

Chiang (2012) explained a practical method conducted by Taiwan Land Office to establish 3D building models that contain property ownership information in each storey in the paper “Data modelling and application of 3D cadastral in Taiwan” presented at the 3rd International workshop on 3D Cadastre held in Shenzhen, China in October 2012. The research was commissioned by the Government of Taiwan to investigate the feasibility of a 3D cadastral system. They used result maps of building survey maintained by Land Offices to integrate building floor plans to construct a 3D property model. This result map consisting of building plan, building location, building area calculation formula, and building attributes data was issued upon completion of construction of a building. Along with the building number, the result map of building survey was used for building registration. Based on the building components in the CityGML standard and the contents in the result maps of building survey, they had designed ten building components to describe the cadastral building model, including “layer”, “3D building”, “3D floor”, “wall”, “door”, “window”, “eaves”, “balcony”, “stairs and elevator”, and “column”. The result map and

construction plan were used to create the detail 3D building model and output as SketchUp format for further 3D data editing. They developed two sub-systems: storey plan vectorization system based on JRE; and basic 3D cadastral building system developed using Apache Tomcat, JRE, and Java 3D to achieve the 3D building model. The final derived basic 3D cadastral building model contained building number data; storey ID; columns; 3D graphical data; 3D surface and textures; floor information; building number and location; storey polygon; view point; and interior walls. OGC standard CityGML was customized for 3D cadastral data exchange format, which includes following classes and subclasses: “3D building” and “3D storey” as subclasses of “AbstractBuilding”; “Roof”, “Polygon”, and “Wall” as subclasses of “BoundarySurface”; “Door” and “Window” as subclasses of “Opening”; and “Columns”, and “Stairs and Elevator” as subclasses of “IntBuildingInstallation”. The 3D model prototype system was developed on SkylineGlobe and utilized zoning maps, topographic maps, orthophotomap, cadastral maps, digital terrain model, and the 3D cadastral building model to integrate with the land and building registration attributes of the National Land Information System for system query and other application. The system was divided into four main function groups such as map query, land service, development plan, and 3D application.

Shen, Lin, and Renzhong (2011) presented a method for building a 3D cadastral management system from survey plans with SketchUp in the paper “Building 3D cadastral system based on 2D survey plans with SketchUp”. This paper defined a cadastral model and builds a prototype 3D cadastral management system corresponding to a 3D cadastral partition of space. Normally 3D cadastral objects have physical space and legal space in a cadastral system, represented by

corresponding mathematical model and legislative definition. However, this paper concentrated only on the geometric representation and topological consistency of a physical object and took little of the legal aspect. The process of building a 3D cadastral management system was divided into four phases: (1) processing of a 2D survey plan; (2) 3D cadastral object construction; (3) topological reconstruction and semantic information joining; and (4) spatial query and analysis. The first phase started with the topological validation of the survey plan complying with the three fundamental characteristics: (1) polygons should be closed, (2) the boundary of polygons should not intersect, and (3) the polygons should not overlap and have no gaps between them. Here, the 2D plan polygon was extruded with the physical height in the Google SketchUp software. Then, the 3D model was linked with other attributes such as geometric description, identifier, legal or property rights and some temporal information by the developed plug-in using Ruby programming language. The prototype developed can do some common functions like general semantic query and display of geographic coordinates. In addition, the topological query can be carried out based on the constructed topologic relationships. With the generic functions of SketchUp, several basic computations for 3D objects can be calculated, such as bottom area, surface area and volume of 3D cadastral units. The prototype can also implement major operations like subdivision of 3D object, and merging of multiple 3D objects. They made a complete use of SketchUp functions for the correct topological structure, spatial analysis as well as to provide a flexible visualization.

Panchal, Khan, Sengupta and Sarda (2011) from Indian Institute of Technology Bombay (IITB) outlined the importance of 3D GIS technologies in establishing and maintaining large-scale, reality-based 3D geo-information services in

their paper, which was a part of project “IITB Smart Campus GIS Grid”. The researchers described how free software like Google SketchUp and standard like CityGML can be used to represent IITB campus data in a 3D environment. The paper presented a semi-automatic approach for modelling the photo-realistic outdoor and basic indoor model of building using Google SketchUp. The shapefile converted to KML format was exported into SketchUp. It was extruded with accurate height information and then modified with architectural drawing (CAD file) in Google SketchUp. The modified model was textured with terrestrial images and finally converted to CityGML format using the SketchUp-to-CityGML plugin. As results, they mentioned that the derived 3D information can be used as an input for the following tasks such as urban planning and economic development, infrastructure management and monitoring, risk assessment system, public safety and security, and virtual navigation system.

CHAPTER III

RESEARCH METHODOLOGY

The conceptual research framework is schematically presented in Figure 3.1. The research mainly consists of four components, namely: (1) analysis, (2) design, (3) implementation, and (4) testing. The components are further divided into sub-component to elaborate the complete workflow from data acquisition to data dissemination. The first component reviews the application of ISO standard LADM and OGC standard CityGML in field of 3D LIS from the available literatures. It also studies the existing land administration system of Bhutan and finalizes the requirement of NLCS. The second component explains the UML designing of the 3D data model according to the finding of the analysis component. Thirdly, the UML design is realized into a relational database. In the final component, the realized database is visually tested. The details of four components of the research methodology are separately described in the following sections.

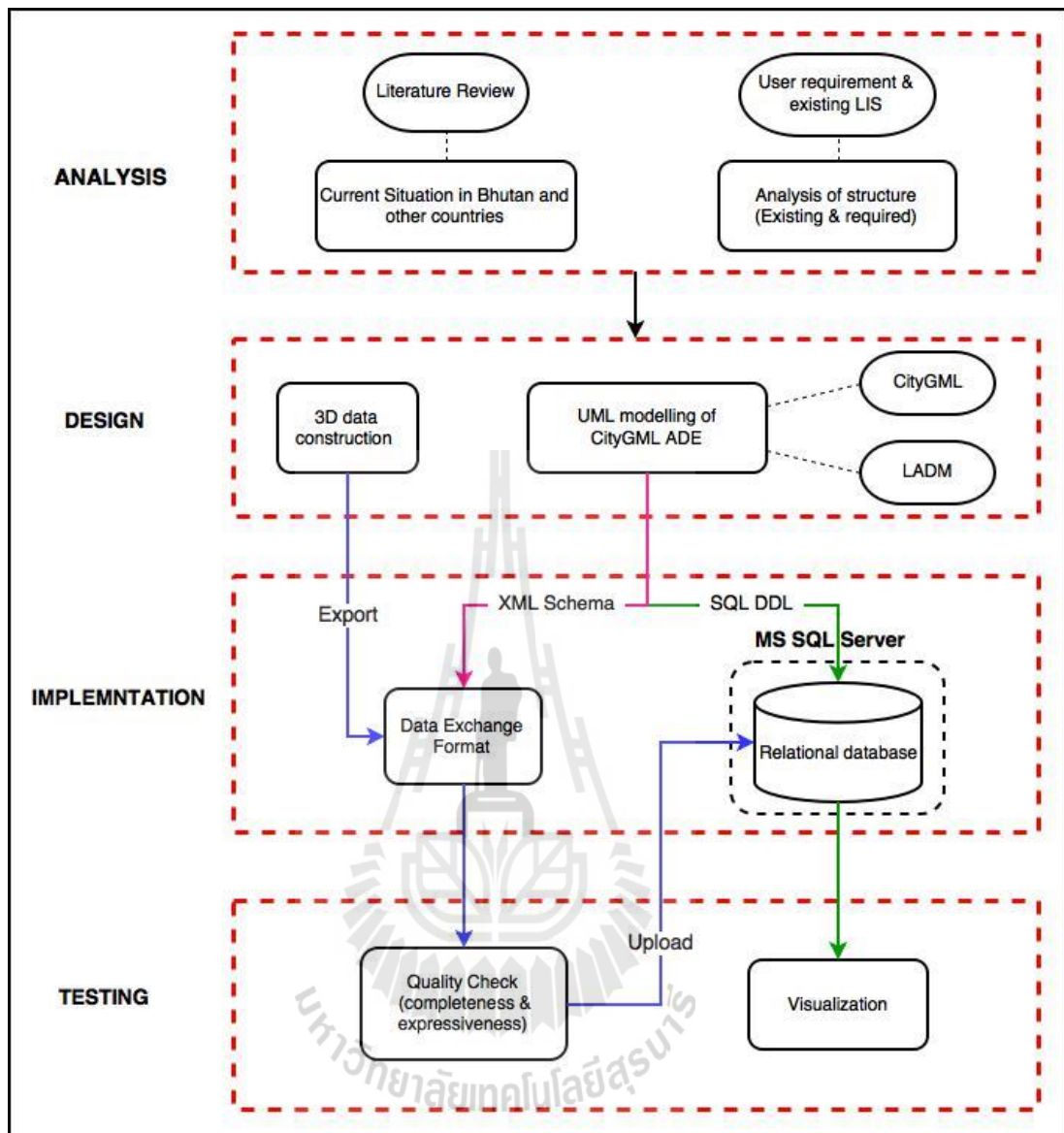


Figure 3.1 The overview of the research methodology.

3.1 Analysis component

This section mainly states the finding of literature review in the field of LIS. It basically covers the four main areas including data collection, research tools, technical review of the existing system in Bhutan, and finally the specification of the new 3D database. These four topics are described individually in the following sub-sections.

3.1.1 Data collection

Literature review shows that the most important part in any sort of research is the data acquisition. Moreover, it is a generic fact that if the data is complete and accurate, then the dissemination and analysis followed will also be accurate. Many researches on 3D data acquisition have been done involving technologies such as LiDAR and terrestrial laser scan (Wang and Sohn, 2011), aerial photograph, and tachymeter. In all the cases, the data from the mentioned technologies are combined with 2D floor plan to construct the detail 3D model, either manually or automatically.

Based on the above finding, 2D floor plan is used in this research to manually construct the 3D model. The data requirement for 3D data construction and modelling are collected from the concerned agencies. The collected data are as follows:

(1) 2D data consists of parcels, buildings, and other topographical data such as transportation, utilities, etc. are collected from the Office of National Land Commission Secretariat (NLCS) of Bhutan in geodatabase format.

(2) Few samples of construction drawings of the building in CAD format are collected through personal contact.

These two data sets are used as an input in the 3D data construction and finally the result are used as a test data to study the 3D data model.

3.1.2 Research tools

Many software packages for building 3D model are available in the market, like AutoCAD 3D Map from AUTODESK, Bentley's Microstation, ArchiCAD from GRAPHISOFT, and SketchUp from Trimble. Shen et al. (2011) stated that all the above software provides various tools and visualization platform for various types of construction and building, but lack topological support needed for the cadastral work. However, SketchUp supports a secondary programming development with Ruby

language to deal with such spatial topologic information, as well as geographic reference and coordinates. In addition, many SketchUp plug-in have been developed and most of them can be freely downloaded online. Furthermore, SketchUp is widely used by the architect community in Bhutan to construct 3D model. So SketchUp is chosen as a tool to construct 3D model in this study.

The next issue to review here is the transformation of SketchUp model to CityGML. For that, two websites: www.geores.de and www.citygml.de/index.php/sketchup-citygml-plugin.html mentioned about a CityGML plug-in for SketchUp. Unfortunately, the plug-in is built for SketchUp version 6 and it is not compatible with the SketchUp 2015 used here. Moreover, in this research the original CityGML schema need to be modified to fit to the cadastral domain, which requires the plug-in to be modified too. As an alternative, Feature Manipulation Engine (FME) from Safe software is here used to transform thematic model prepared in SketchUp model to CityGML model, based on the research work of Wate (2014). FME offers a solution for data transformation issues, supporting over 275 different data formats (Stoter et. al, 2013). It supports transformation between formats and coordinate systems and also transforms data models and schemas.

Several commercial software for UML modelling such as Visual Paradigm (VP-UML), Enterprise Architect (EA), and Microsoft Visio are available. Andrade et al. (2013) developed a complete workflow from conceptual and abstract model to a relational database using VP-UML. They stated that the choice of the VP-UML resides in its capability to execute the whole modelling process, from the construction of the UML diagram until its implementation in a database. The choice of Microsoft Visio was implemented by the Swesurvey member during the project on

implementation of the 2D cadastral database in NLCS, Bhutan. Hespanha (2012) presented the specification of UML in a relational database, where all the original modeling was done in EA and exported to Eclipse UML for further processing. In addition, EA is widely used in the world of UML modeling. A variety of CityGML ADEs was modelled using the EA tools in various fields such as property taxation (Cagdas, 2013), management of indoor facilities (Kim, Kang, and Lee, 2013), utility networks, noise modelling, solar potential, energy efficiency of buildings as well as national implementations in the Netherlands (van den Brink, Stoter, and Zlatanova, 2012), Germany or Bahrain. Therefore, EA is chosen as a tool to develop UML model of CityGML-ADEs in this study, as well as its realization of relational database.

Based on the above finding and its availability, the following tools which are used to fulfill the objectives of the research are here again summarized.

(1) SketchUp: Trimble SketchUp 2015 is used mainly for editing and modeling of 3D objects. SketchUp offers a module called Ruby, which is object-oriented programming language, where plug-in, can be programmed to automate some functions of 3D modelling.

(2) Feature Manipulation Engine (FME): The conversion tool FME of Safe Software Inc. is used to convert the 3D SketchUp model into CityGML model, which is finally loaded in the database maintained in MS SQL Server.

(3) Enterprise Architect (EA): Enterprise Architect version 12.0 of SPARX Systems is used as a UML CASE tool to create conceptual and logical models of CityGML-LADM ADE in the form of class diagrams. The UML model is converted to DDL format to realize a relational DBMS in MS SQL Server.

(4) **ShapeChange:** ShapeChange is a free Java tool that can construct application schema according to ISO 19109 standard from a UML model (Shapechange, 2015). A UML model stored in an EA project (EAP file) can be directly processed by ShapeChange using the Java API of EA.

(5) **ArcGIS:** ESRI ArcGIS Version 9.2 is used to edit the existing data and to add the height of the buildings. It is also used to edit the elevation of the building footprint. Here, the elevation of the existing data is based on the Geoid 1980, which is commonly called the mean sea level (MSL) height.

(6) **XMLSpy:** Altova XMLSpy 2016 is XML editor for modeling, editing, transforming, and debugging XML-related technologies. It offers graphical schema designer, a code generator, file converters, debuggers, profilers, full database integration and supports for XSLT, XPath, XQuery, WSDL, SOAP, XBRL, JSON, and Office Open XML (OOXML), plus Visual Studio and Eclipse integration, and more.

(7) **Viewer:** Many viewers for 3D model are freely available including 3DGML3 3D viewer application from Aristoteles, LandXPlorer CityGML Viewer 2009 developed by the company Autodesk, and CityGML and IFC viewer application FZK Viewer. In this study, IFC viewer application FZK Viewer is used for visualizing and testing the 3D model.

3.1.3 Analysis of existing system

The cadastral system of Bhutan is a parcel-based land information system, governed by NLCS. It comprises a land registration system and a cadastral

registration as key components to house the legal information and physical properties of land parcel respectively. In year 2008, under the Royal Command, NLCS carried out nationwide cadastral resurvey with the latest technologies (such as GPS, Total Station). During the resurvey the details such as parcel boundary, road, building, river, utility, etc. were surveyed and the final data were uploaded in the cadastral database maintained in MS SQL Server. The cadastral database contains the coordinates (plane and height) of each and every point of the parcel along with other topographical information, whereas registry database stores legal information like details of landowner, land type, area, mortgage information, etc. in the form of table. A brief description about the two databases is highlighted below.

A combined Topo and Cadastral Spatial Database exist in NLCS for the management, dissemination, and archiving the geospatial data in the country. The feature list contained in the data model represents all topographic data and cadastral data. All data are stored in ArcGIS geodatabases in geographic coordinates in DrukRef03, the new reference system of Bhutan. The combined model is divided into sub-models, which are the datasets in the final geodatabase (Table 3.1). Among the nine datasets (sub models) of NLCS data model, the relevant dataset ‘Building model’ is shown in the Figure 3.2. It consists of different type of buildings such as institutional, private, religious, industrial, and community. The class names are suffixed with alphabets “P”, “L”, “P”, and “T” to differentiate details as point, line, polygon, and text respectively.

Table 3.1 Content of Cadastral data model of Bhutan (Swedesurvey AB, 2003).

Model name	Content
Pol & Adm Boundary Model	All types of boundaries plus plot polygons.

Buildings Model	All kinds of community, private, religious, and industrial buildings.
Control Model	Plane and height control points.
Hydrography Model	River, lakes, glaciers, and manmade things like channels.
Land cover Model	The main Land cover areas like forest, food crops, recreation etc.
Quality Model	An abstract class to group quality attributes to be inherited from other models.
Terrain Model	Contour lines and other types of terrain lines and points.
Transportation Model	Roads, paths, ropeways, airports and symbols to these.
Utilities Model	Electrical, water, sewage and telecommunication lines and points.

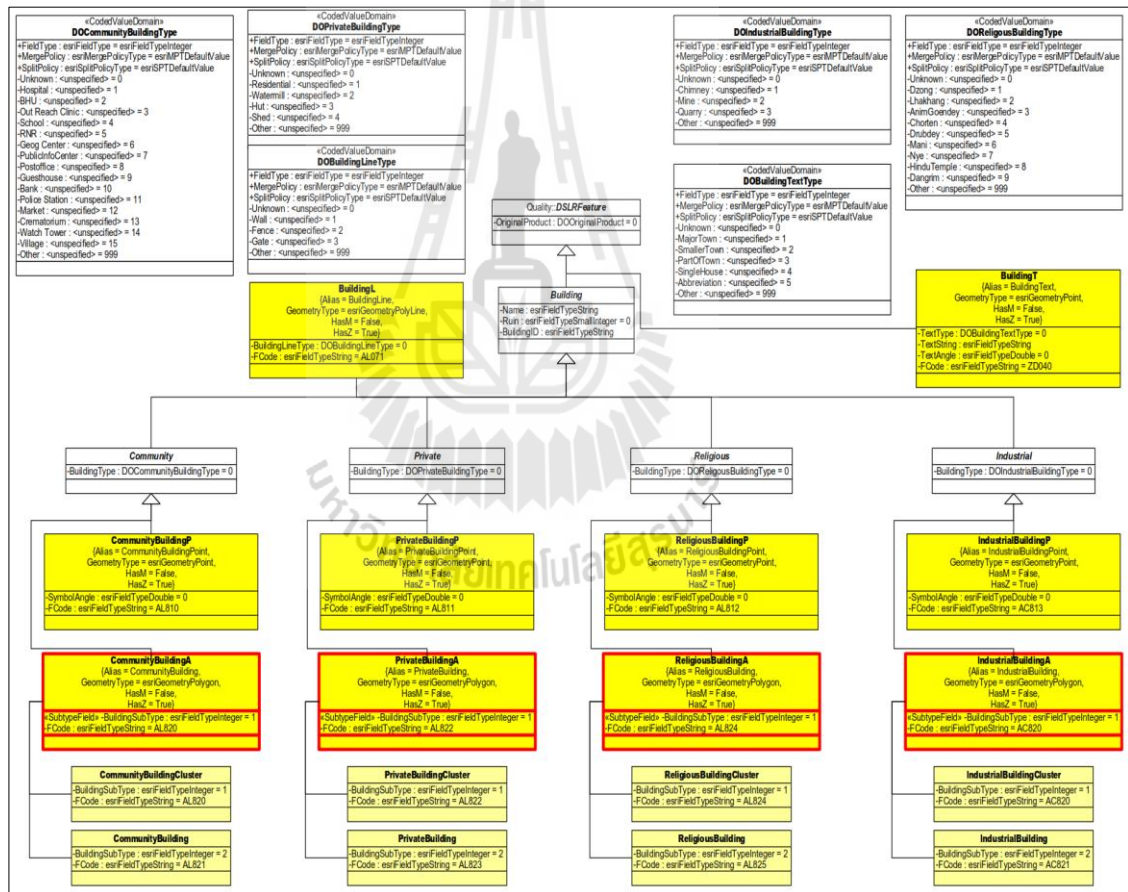


Figure 3.2 Content of Building Model of Bhutan (Swedesurvey AB, 2003).

All features defined in the cadastral database are assigned a feature code to avoid input of different code for the same object. A code list simplifies a data input

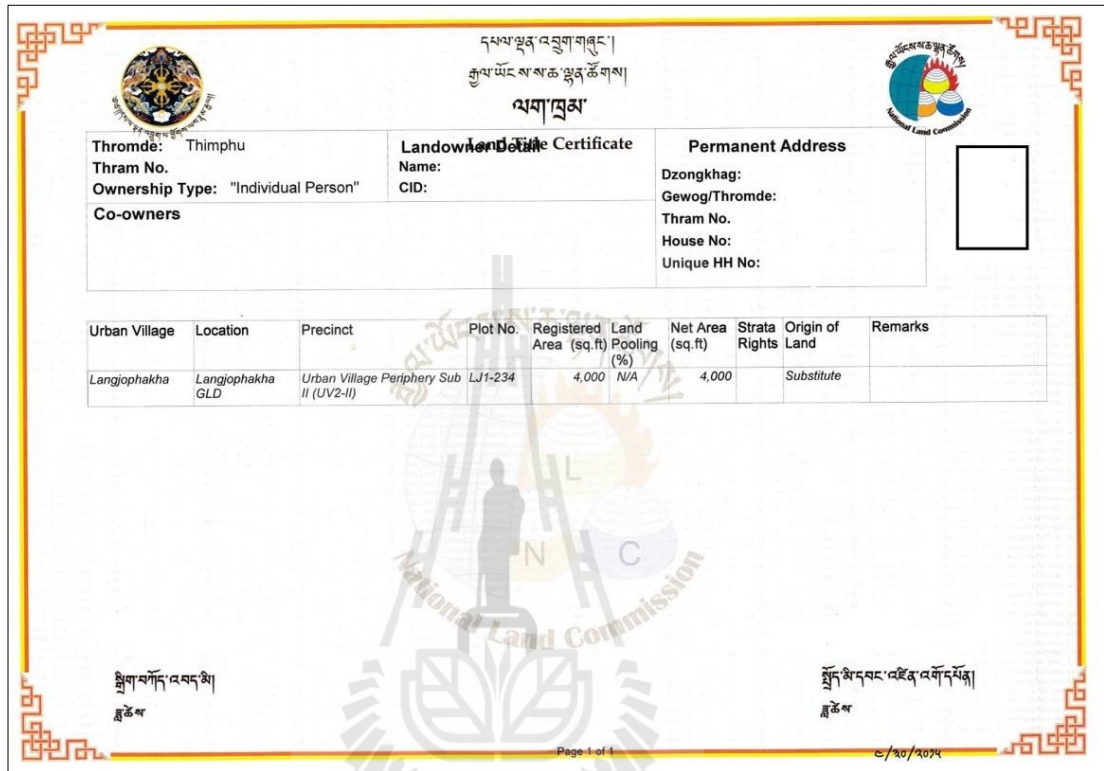
as well as give a definition of the features. An extract of NLCS Feature Code is shown in Table 3.2.

Table 3.2 NLCS feature code for Building Model.

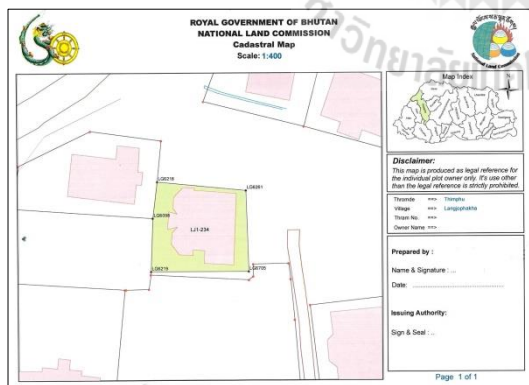
Sl. No	Layers/Group	Description	Code
1	BuildingL	Fence	FENCE
		Concrete Wall	WALL
		Gate	GATE
2	PrivateBuildingA	Building/House	BLDG
		Ruin House	RUINH
		Hut	HUT
		Animal Shed	SHED
3	PublicBuildingA	Government Quarters	GQTR
		Government Institutes	GINST
		Government Hospitals	HOSP
		Government Office	GOFF
4	ReligiousBuildingA	Chorten	CHTN
		Dangrim	DANR
		Lhakhang	LHAK
		Dzong	DZNG
		Religious Institute	RINST
		Hindu Temple	TMPL

Land registry database mainly contains the information of the landowners, land parcel and rights, responsibilities, and restrictions related to it. The database consists of tables such as Thram (registry), Plot, District, etc. associated with each other through the use of primary key (PK) and foreign key (FK) pairs. The main link between the landowner and land parcel within the registry database is a registration number, which is a numeric character generated uniquely within a sub-district. The database has the capacity to record history on the registry and also on the parcel. A land registration interface was developed in Microsoft Access to automate the land registration process and further a browser interface was developed which integrated the

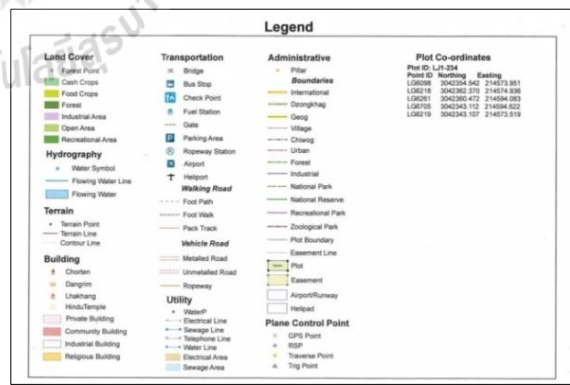
map and registration data for viewing purposes. Figure 3.3 shows an example of title certificate issue in A4 format by NLCS to individual owner.



(a)



(b)



(c)

Figure 3.3 An example of title certificate issued: (a) Certificate, (b) Map, and (c) Legend.

A unique Plot Id Number (PIN) serves as the functional linkage between the cadastral map and the registry database. It is maintained in the cadastral map to

define a plot anywhere in the country. The same unique PIN is also maintained in the registry database. A unique parcel identifier is adopted for every sub-district, which is composed of alpha characters that abbreviates name of the sub-district and a running numeric character which is unique within every sub-district for rural areas (e.g. MEW-1012). For the cities, the following unique identification is adopted: (1) For cities, the PIN is composed of two letters followed by one number that that abbreviates the block in the city and a running numeric character which is unique within that block (e.g. *UVI-100 or UV11-100*), and (2) for core areas, the PIN is composed of the word CORE that abbreviates the city kernel and running numeric character which is unique within that kernel (e.g. CORE-5 or CORE-15).

3.1.4 Requirement and specification of new 3D LIS

The requirement of the new 3D system is to realize the problem of registration and maintenance of the physical data of apartment. Based on the definition of requirement, the specification of the new system is firstly formulated. The required specifications of the database is then compared with the chosen model LADM and CityGML to define classes, attributes, relationship, and constraint needed to model the building database.

Similar to the 2D cadastre, 3D cadastre defines the 3D legal space occupied by the buildings and its corresponding 3D physical space. Literature review shows that LADM can support both 2D and 3D representation of cadastral objects, but only the legal space. As mentioned before, LADM requires an external class to store the physical data. This external class can be realized using the modules of the CityGML. Gózdź et al. (2014) confirmed that CityGML module can be adapted to land administration domain, particularly for supporting the spatial concepts required for

cadastral systems. However, it does not contain any features describing the legal information about spatial objects. Therefore, the classes of the LADM, which support the storage of legal space of a 3D property was combined with the classes of CityGML, which covers the physical concept.

The next step is to define the various cadastral objects needed to define legal aspect of 3D registration. Zulkifli et al. (2014) defined the cadastral objects included in the Malaysian LADM country profile as building and building parts (all in 3D within a single lot), land parcel (with house no more than 4 floors within a single lot), which can be refined with parcel unit, accessory unit, and (limited) common property unit including support for provisional and multilayer/underground aspects (Figure 3.4).

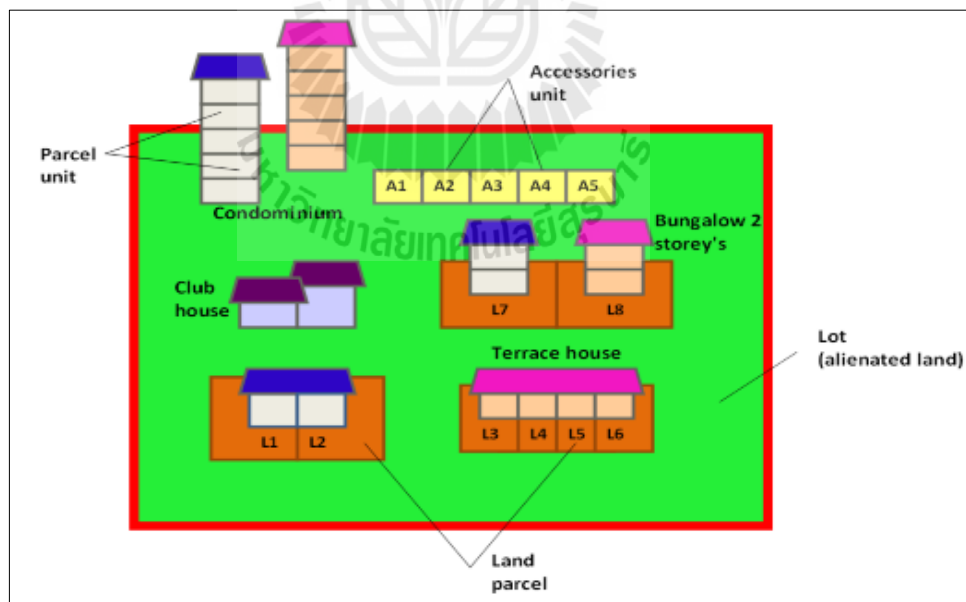


Figure 3.4 Various cadastral objects related to strata titles in Malaysia (Zulkifli et al., 2014).

The legal aspect of the 3D registration of Malaysia is here mentioned to give some idea about cadastral objects. Bhutan does not have a legal document to

clearly define the various cadastral objects. Therefore, based on the literature review and from common observation in strata development in Bhutan, the following cadastral objects are defined: (1) building, (2) apartment unit, (3) common areas (such as staircase, parking area, and roof), and (4) accessory unit (such as garage, storeroom).

Figure 3.5 shows an example of drawing in strata tile.

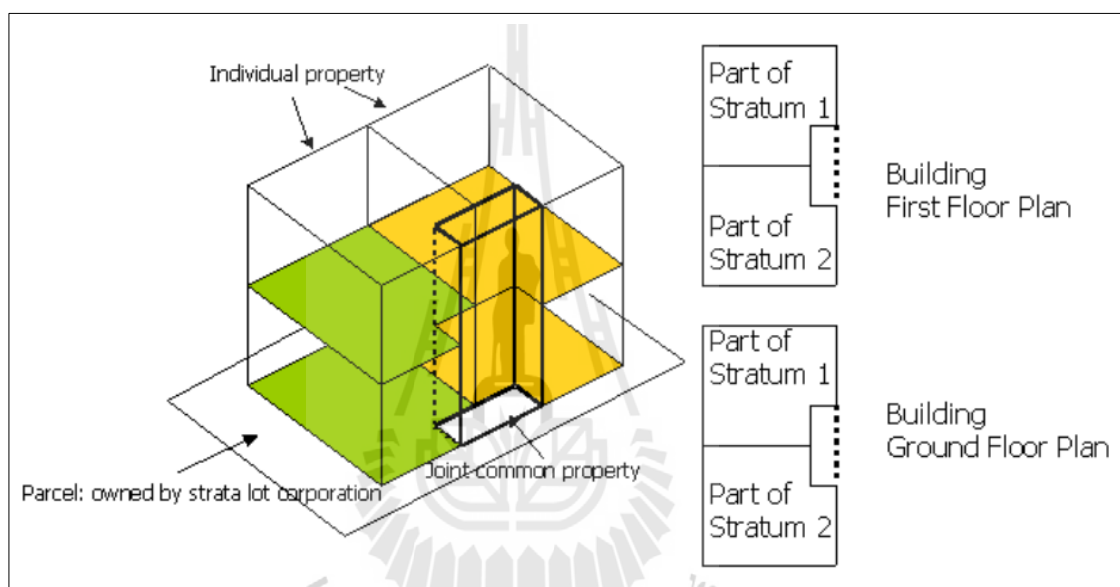


Figure 3.5 Example of drawing in strata title (Stoter, 2012).

Various building components are required to define the legal aspect related with a building. Chiang (2012) defined ten building components to describe the cadastral building model of Taiwan based on the building components in the CityGML standard, including “3D building”, “3D floor”, “layer (room/unit)”, “wall”, “door”, “window”, “eaves”, “balcony”, “stairs and elevator”, and “column” as shown in Figure 3.6. Similarly, these components can be considered to define the both the legal and physical properties of a building in Bhutan. However, the component such as eaves, which is not common in Bhutanese architecture, can be rejected. Additionally

components floor and column are not considered based on the priority for this study. Instead the floor can be added as an attribute to component layer. Based on finalized specification, the various building components is constructed in SketchUp as well as added in the CityGML ADE to be realized into database.

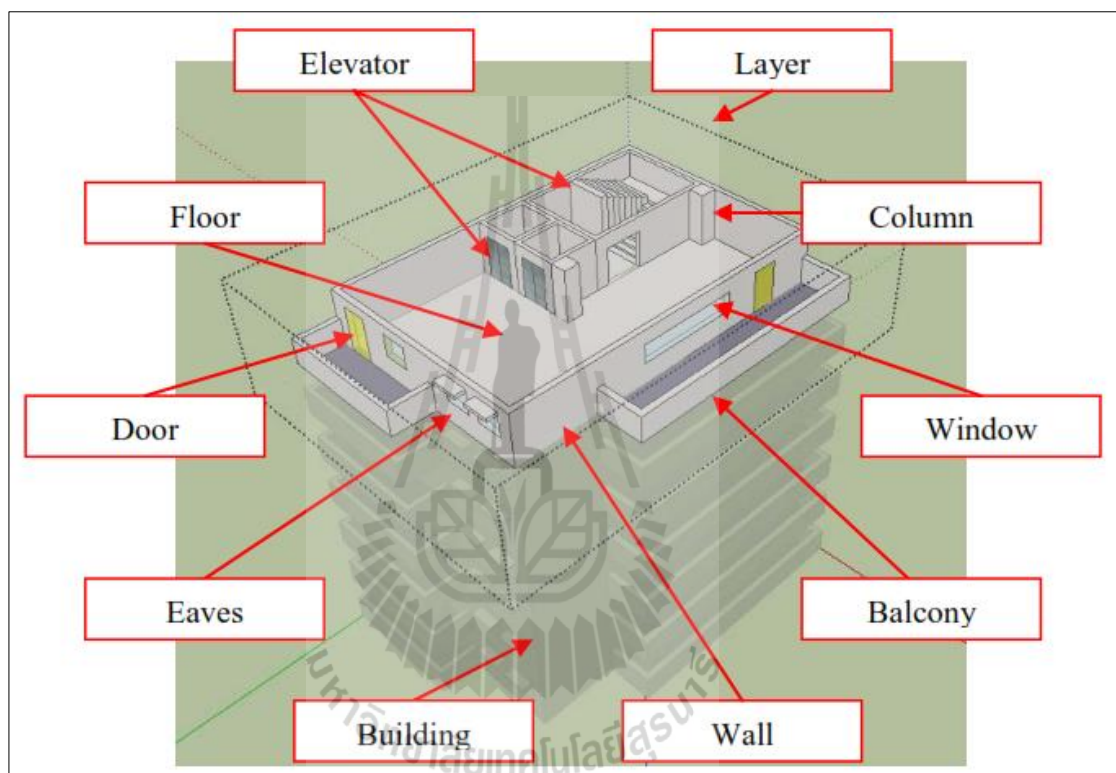


Figure 3.6 Illustration of main components of cadastral building model (Chiang, 2012).

Another aspect of 3D cadastre to be considered is the definition of the boundary of the 3D properties. From legal point of view, when 3D properties contain building boundaries, the location of those boundaries must be defined by stating whether those boundaries lie along the “Interior face”, “Exterior face” or the “Median” of the relevant physical structure (Aien et al., 2013). For example in Victoria, Australia, definition of interior face often lies along the interior face of any wall, floor (upper

surface of elevated floor if any), ceiling (underside of suspended ceiling if any), window, door or balustrade of the relevant part of the building (Figure 3.7). Selecting the interior face as the parcel boundary generally means the structure of the relevant wall, floor, and ceilings is to be contained in common property. The other option can be the selection of exterior face of the building component as the legal boundary of a property, which means the legal space of a building is equal to its physical space.

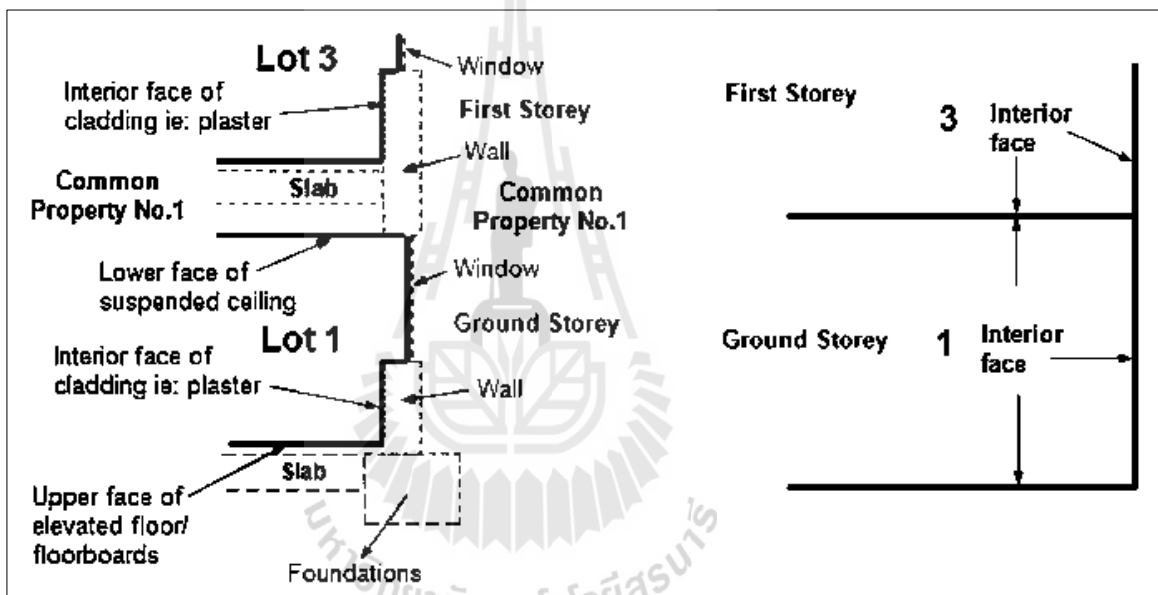


Figure 3.7 Illustration of the definition of interior face in example of Victoria, Australia (physical view) (left) and its representation in the subdivision plans (legal view) without depicting the structure (Aien et al., 2013).

The definition of boundary of 3D property (ownership space) should be supported by the legal policy of that country. Presently Bhutan does not have legal documentation to support this definition. However, for this study interior faces of an apartment as defined in the above example of Victoria, Australia is here considered as legal boundary.

3.2 Design component

Design component is divided into two parts: 3D data construction, and UML modelling of CityGML ADE. The two design works are elaborated in the following section.

3.2.1 Construction of 3D data

The workflow for the construction of 3D building model is shown in Figure 3.8. The building footprint and floor plan is used as an input for the construction of 3D building model in SketchUp software manually. This software supports dxf and kml format, therefore the building footprint in shapefile is firstly converted into dxf format using ESRI ArcMap software and then floor plan is directly imported to SketchUp. The process is divided into three steps:

- (1) Firstly, the floor plan is georeferenced with the building footprint.
- (2) Secondly, the georeferenced floor plan is modified with the measurement details mentioned on the floor plan to specify the details such as walls, floor, ceiling, window, door, etc.
- (3) Finally, the different features of the modified building are assigned to its specific layers.
- (4) The final Sketchup model is translated to CityGML model using FME.

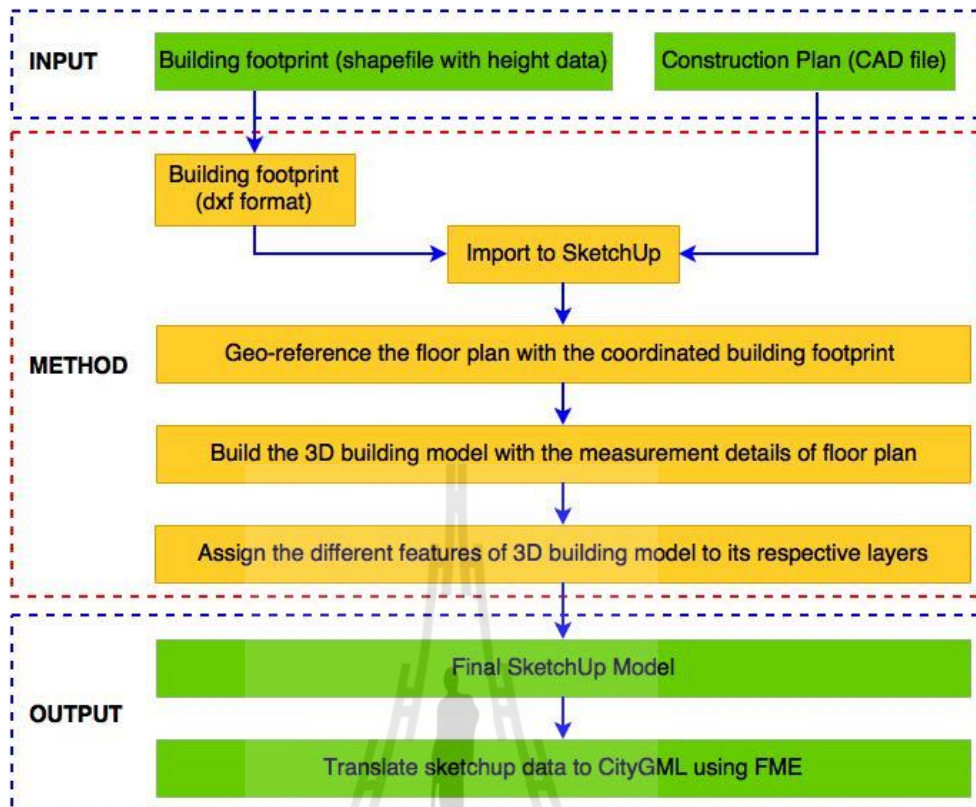


Figure 3.8 Workflow for the construction of 3D data.

3.2.2 UML modelling of CityGML ADE

The focus of this section is to realize a CityGML ADE for the 3D cadastre in Bhutan. Since CityGML is extended for cadastre in this study, henceforth CityGML ADE is declared as Cadastre ADE.

CityGML version 2.0.0 realized in year 2012 freely available in the website <http://www.citygml.org> is used in the study. CityGML consists of one core module and thirteen thematic extension modules. However, the study does not require all the extension modules to be included in the ADE. Herewith, only the core and building module of CityGML are used and extended with the required additional classes. CityGML building module has the feature classes to represent room, building furniture, boundary surface, and opening in indoor space. However, it lacks the classes

which define the legal aspect of 3D properties needed in cadastre. So a conceptual model is realized supporting the description of both legal and physical space of buildings.

Cadastre ADE is modelled using an UML class diagram which is one of the most used modelling languages by standardization bodies dealing with Geoinformatics (van den Brink et al., 2013). CityGML specification does not provide rules or guidance to model an ADE using UML (van der Brink et al., 2012). Therefore, the approach adopted by the van der Brink et al. (2013) in the paper “UML-based approach to developing a CityGML Application Domain Extension” is here used to extend the application of CityGML for 3D cadastre. This practice is also later adopted by OGC as the best practice for modelling an ADE of CityGML in UML. Figure 3.9 shows an example of CityGML-IMGeo ADE of the Netherlands.

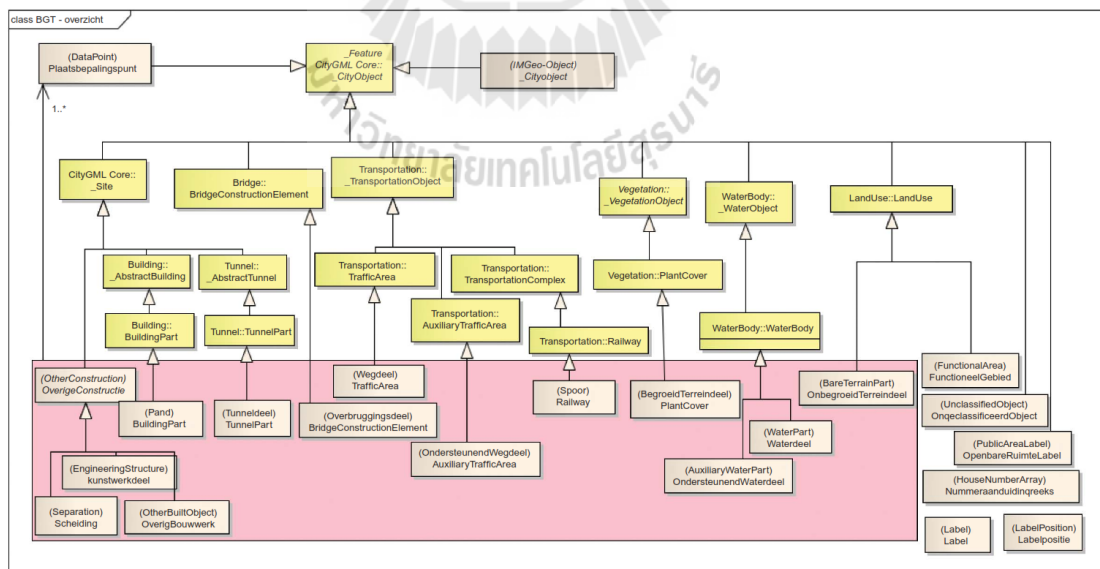


Figure 3.9 An overview of CityGML-IMGeo ADE (van den Brink et al., 2013).

Steps for generating the Cadastre ADE schema are as follows:

(1) Since CityGML is only available as XML schema from OGC, UML model has to be recreated in the modelling tool Enterprise Architect, based on OGC (2012). However, the UML model of the CityGML is available in the website <https://github.com/opengeospatial/CityGML-3.0/tree/master/Model> and is collected.

(2) Conceptual mapping is performed between required classes of this research and CityGML classes to identify the corresponding classes. This process lists all the correspondences, mainly in the class level.

(3) Decide which subclasses to be extended. There are two types of correspondence. Either the class is semantically the same as the corresponding CityGML class, and only adds properties, or the class is semantically a subclass of the corresponding CityGML class. In the first case, the classes get the same name as the CityGML class they is extended with stereotype <<ADEElements>>.

(4) Definition of code lists of Bhutan. The code lists is maintained in the UML model and generated as XML structured code lists using ShapeChange tool, which allows generation of SKOS-encoded code lists with stereotype <<code list>>.

(5) Finally, the XML Schema (GML application schema) is generated from the ADE defined in UML using the Java tool ShapeChange.

The above procedures only define the schema for exchange and storage of 3D data. This schema needs to be transformed to a database structure to be able to store the huge amount of data generated through the 3D data construction. This matter is elaborate more in the next component.

3.3 Implementation component

Implementation component of this research consist of the two parts: data exchange format and specification of relational database, which are explained separately in the following sections.

3.3.1 Data exchange format

A data exchange format is needed for storage and exchange of data between systems or when sharing data between agencies. There are some data transfer standards available in the market such as LandXML, BIM, CityGML, etc. However, instead of using these standards, a data exchange format is here generated as XML schema, which has been explained in last part of previous Section 3.2.2.

3.3.2 Specification of relational database

This section explains the design and specification of realization of a relational database. Basically it is mapping of the UML model into a relational database where the classes, attributes, and relationship that exist in the UML model is realized in the form of tables, columns, and cardinality in the chosen relational database. In this research, the relational database is realized in MS SQL Server 2008.

During the process, many design decision has to be taken to develop the database schema. A class in the UML model normally corresponds to table with same name in the database model. Additionally, there are also views, tables for code lists, and additional tables for representing relationships in case of a many-to-many relation between two classes. There are various types of relationships to define relation between tables in a database such as association, inheritance, and aggregation. Sparks (2015) stated that inheritance is the most problematic relationship to define in a relational database and there are three ways to handle it in a database: (1) defining each class

hierarchy as a single table, (2) defining each class as table (including inherited attributes), and (3) defining each class in a hierarchy as table containing only its attributes. Another point of attention is defining a proper Primary Keys (PK), Foreign Keys (FK) and indices (usually a B-tree) in order to implement relationships between objects in an effective manner.

Since EA supports comprehensive functionality for modeling database structures (Sparx Systems, 2011), all the modelling phase from the conceptual model to the DBMS realization is done in EA. This section is continuity to the Section 3.2.2, whereby the defined class model is used as an input to realize a DBMS. The class model resulted from Section 3.2.2 is converted to physical model in EA. In the final step, the SQL script is generated and loaded into the DBMS (Sparx Systems, 2011).

3.4 Testing component

Some samples of 3D building model are constructed to test the data transfer format and the 3D geodatabase. To test the designed data model, the implementation for the Cadastre ADE is checked in following two ways as suggested by Kim et al. (2013):

- (1) Validations of the XML schema file (Section 3.3.1) using XMLSpy.

- (2) Verification of expressiveness and correctness of sample data by visualization software. Herein, FZK Viewer is used to visualize the developed 3D building data.

CHAPTER IV

RESULTS AND DISCUSSIONS

Main results of the research included (1) specification of new 3D building database; (2) construction of 3D building model; (3) UML modeling of CityGML ADE for 3D cadastre; (4) data exchange format development; (5) building database; and (6) testing are reported in following sections.

4.1 Specification of new 3D building database

This section studies the classes of CityGML which defines physical properties of a building and identifies those additional properties required to define legal space of a building. CityGML building module has the feature models to represent room, building furniture, boundary surface, and opening. However, the existing model needs to include detailed feature classes to represent the characteristics of 3D cadastre. Therefore, the question “what are the properties or attributes that could be added in order to have all legal details into the class?” need to be answered (Dsilva, 2009).

The finalized additional properties for 3D cadastre representation were added to CityGML either as a class or as attributes to an existing class. These two methods of extension of CityGML are briefly explained below.

4.1.1 New attribute definition

Dsilva (2009) described the properties that are needed to describe legal information in a building as building owners, the building registration number or the

building number, the parcel number of the parcel to which the building belongs to, and the type of the building. Though all the mentioned properties are necessary to add legal information in CityGML, here only building number, which can link a physical building to an owner, was added along with the parcel number on which the building stands. The other properties such as building owners and building registration number were found unnecessary to be added. These attributes can be defined in the registry system, which is the normal practice followed by NLCS of Bhutan. The attributes building number was defined with attribute name “buildingNo” and parcel number as “plotNo”, both as a string datatype. These attributes were added to AbstractBuilding class of CityGML with stereotype “<ADEElement>”, which suppress the class from generating an instance in the XML schema of Cadastre ADE.

More attributes such as roomID and buildingInstID was assigned to the class Room and BuildingInstallation respectively to define a unique identifier. Through this identifier the physical objects defined in the above two classes was linked to its corresponding legal components.

4.1.2 New class definition

Table 4.1 shows the new classes added to CityGML to define the legal properties of and related to a building.

Table 4.1 List of new feature classes defined in CityGML.

SuperClass	Class	Subclass	Remarks
SpatialUnit	LegalSpaceBuildingUnit	LegalSpaceBuilding	➤ Legal space of a building,
		ApartmentUnit	
		AccessoryUnit	➤ Defined as subclasses of CityObject class
		CommonPropertyUnit	

New feature classes listed in the above table were added to CityGML building model. Apartment class defines a volume representing the legal space within a building. The common property class identifies the structures within a building belonging to all the registered apartment owners. Whereas the accessory unit class defines the structures which are not attached to the building, but belong to an individual or a group of apartment owners. The concept of classes SpatialUnit and LegalSpaceBuildingUnit were borrowed from the ISO standard LADM. As mentioned in the literature review SpatialUnit class defines all the legal 2D and 3D properties such as building, land parcel, utilities, tunnel, and etc., while its subclass LegalSpaceBuildingUnit outlines all the legal spaces related to building.

The graphical composition of the new classes and attributes added to the CityGML building model is illustrated and explained in the Section 4.3.

4.2 Construction of 3D building model

This section describes the result of construction of 3D building model. 3D data construction consists of two phases included 3D building modeling in SketchUp and its translation to CityGML format. The results from these two phases were graphically illustrated and explained separately in the following subsections.

4.2.1 Modeling of 3D building in SketchUp

Figure 4.1 displays the input: building blueprint (CAD) and building footprint (.gdb) used for construction 3D building model in Trimble SketchUp. Building footprint provided the georeferenced position of the building with coordinated building corners, whereas building blueprint specifies the detailed measurement of the building structures.

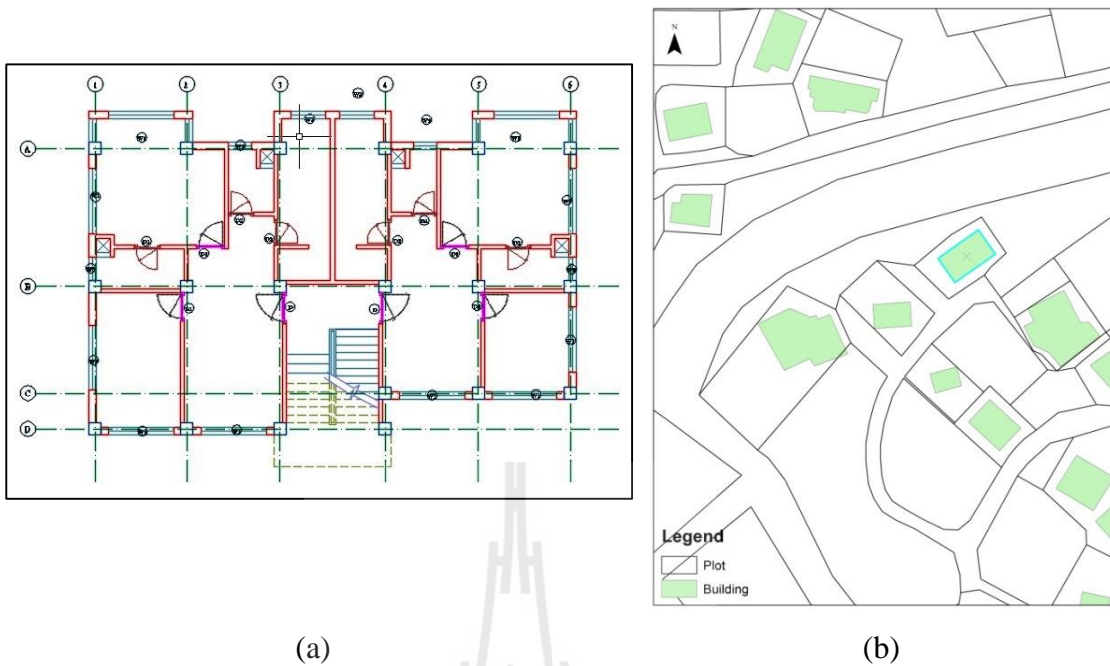


Figure 4.1 Example of the input data: (a) blueprint of building floor plan (CAD) and (b) the building footprint from cadastral data (.gdb).

The result from the first phase of data construction in SketchUp is displayed in Figure 4.2. It consists of a 3D building model with its details segregated as different layers. The window layer on 2nd and 3rd floor was turned off to display the inner view of the building model. This segregation into different layers helps in translating individual component of a building to its respective classes in CityGML.

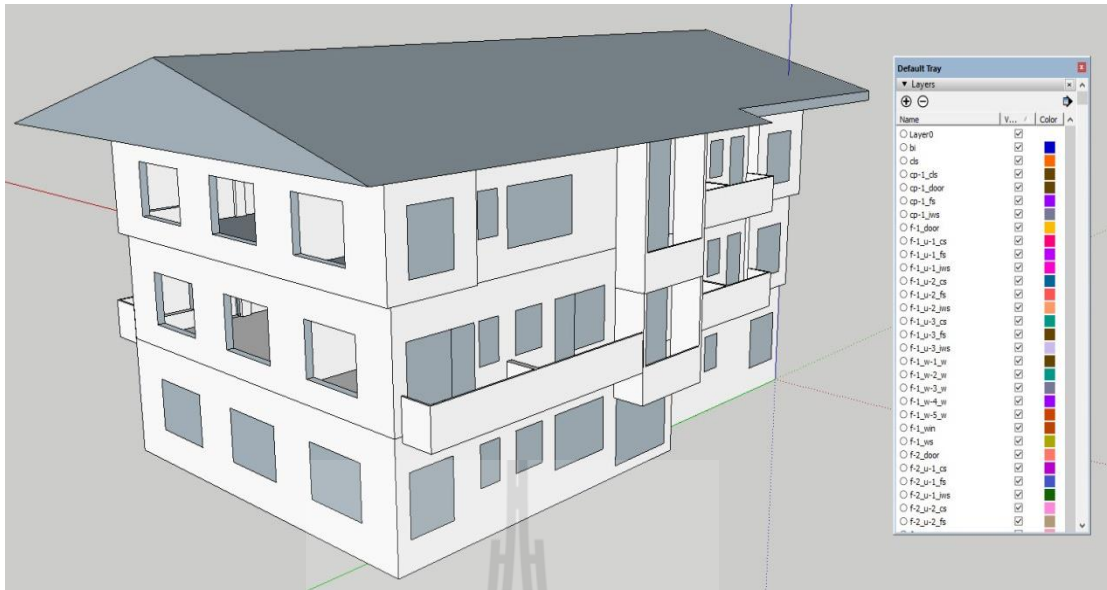


Figure 4.2 Example of output of 3D data construction as 3D building model (.skp) under Trimble SketchUp.

Here it was found that Trimble SketchUp software can efficiently construct 3D building model with its specific details. However, it is a time consuming task and work becomes more tedious with increase in the details. Therefore, the decision on the detailing should be decided in advance before starting 3D building construction.

4.2.2 Translation of SketchUp to CityGML in FME

This section explains the result of the translation of layers in SketchUp to its respective classes in the CityGML. Thematic model prepared in Trimble SketchUp was transformed into CityGML using FME. SketchUp to CityGML conversion with SketchUp Reader and CityGML Writer was developed using sequence of built-in FME transformer tools for ten LOD1 and one LOD4 buildings.

The FME workbench containing SketchUp reader, CityGML writer and transformers is shown in Figure 4.3. This translation consist of two input

SketchUp files: one file with ten building footprints and another with a 3D building model (Figure 4.2). The input building footprints were extruded with the height of the building to create LOD1 building models. For the second input, individual layers defined in the SketchUp were translated into its respective feature classes such as RoofSurface, WallSurface, GroundSurface, InteriorWallSurface, BuildingInstallation, Window and Door with Building as its parent feature class. The workflow of main process and individual feature class translation are separately illustrated in Figures 4.4–4.13 for clearer understanding.



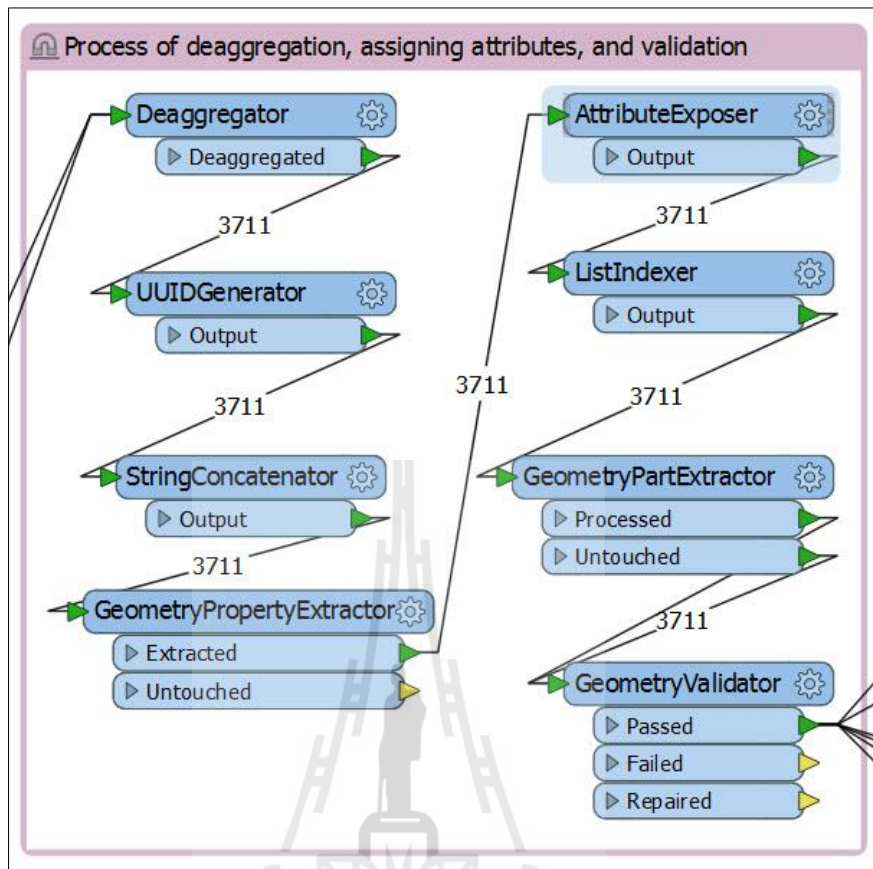


Figure 4.4 Workflow of SketchUp data preparation before writing into CityGML classes.

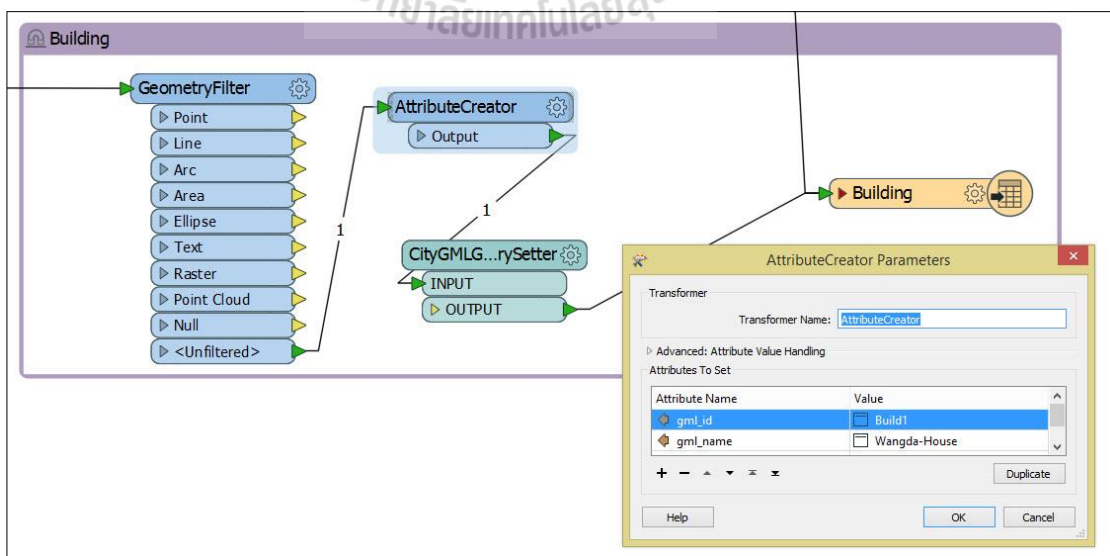


Figure 4.5 Translation of LOD4 building.

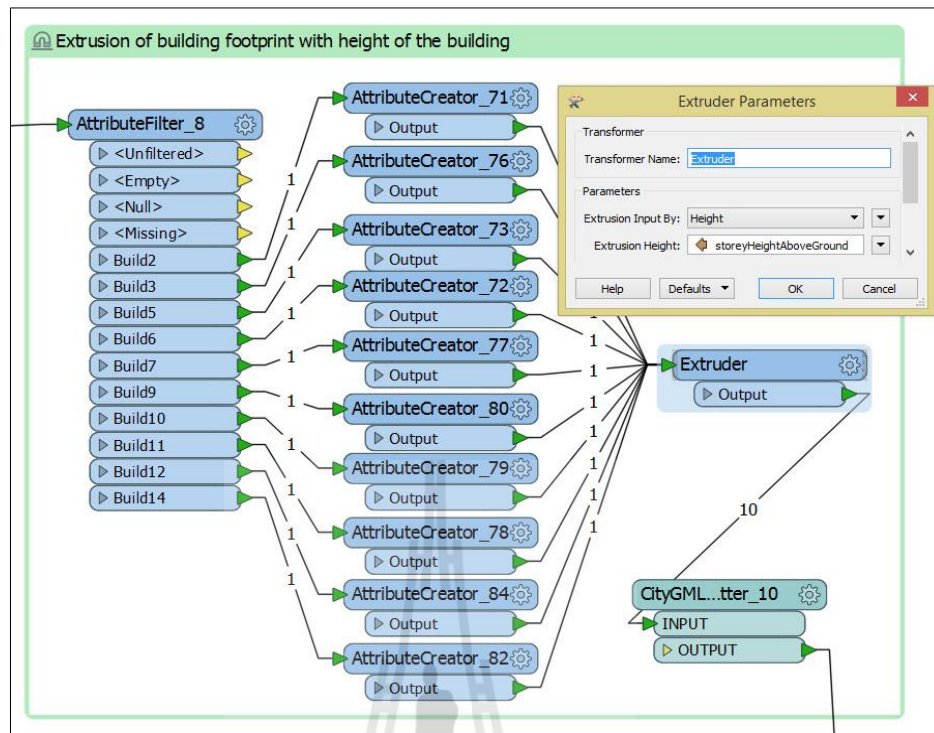


Figure 4.6 Translation of LOD1 building.

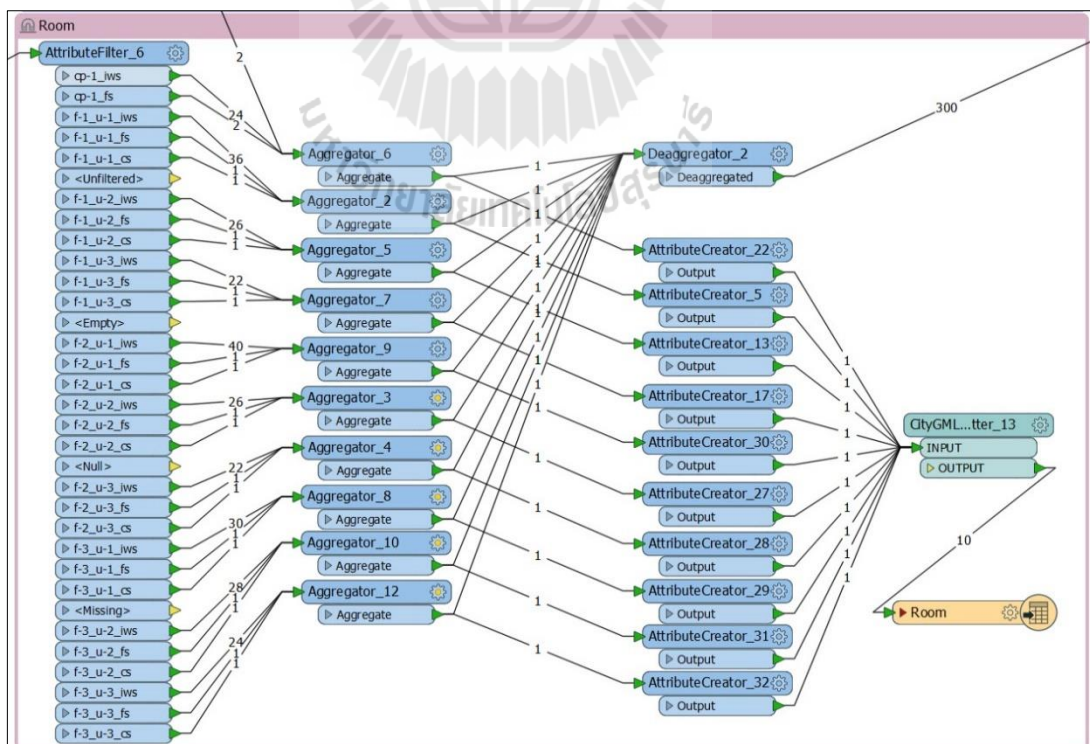


Figure 4.7 Translation of room.

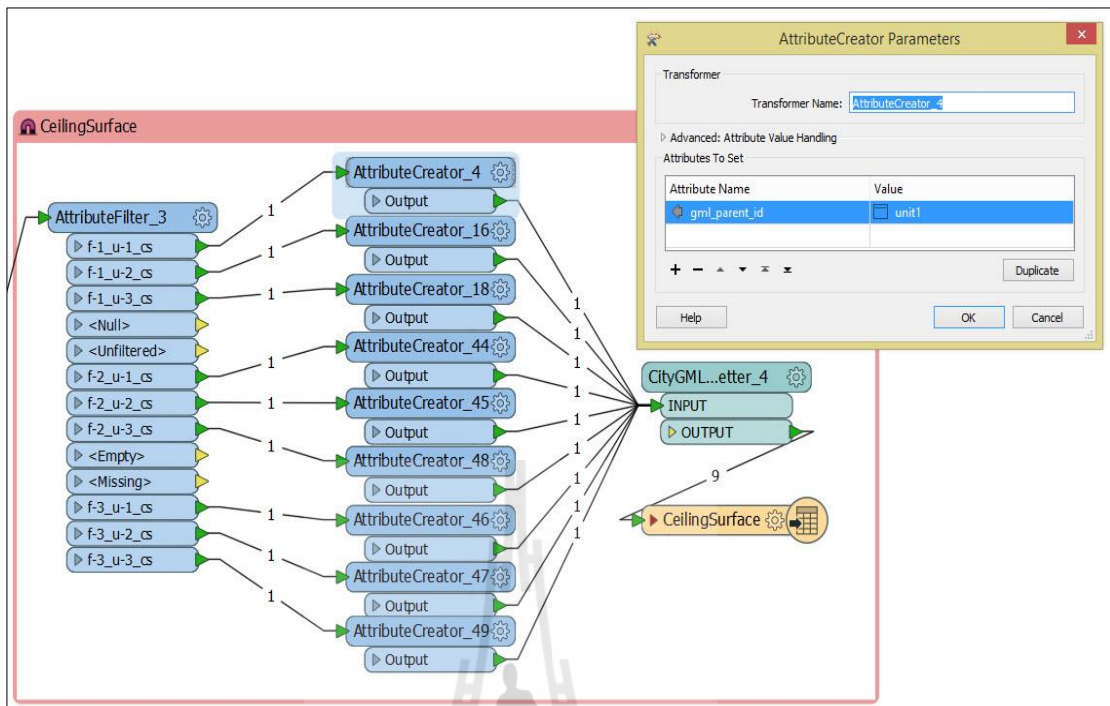


Figure 4.8 Translation of ceiling surface.

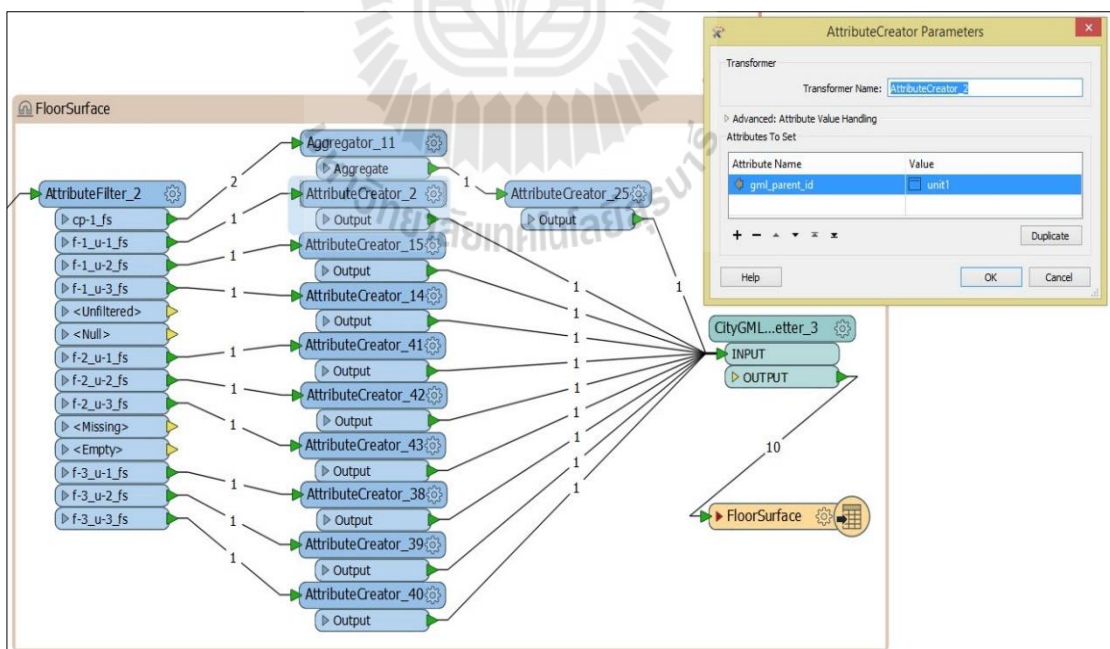


Figure 4.9 Translation of floor surface.

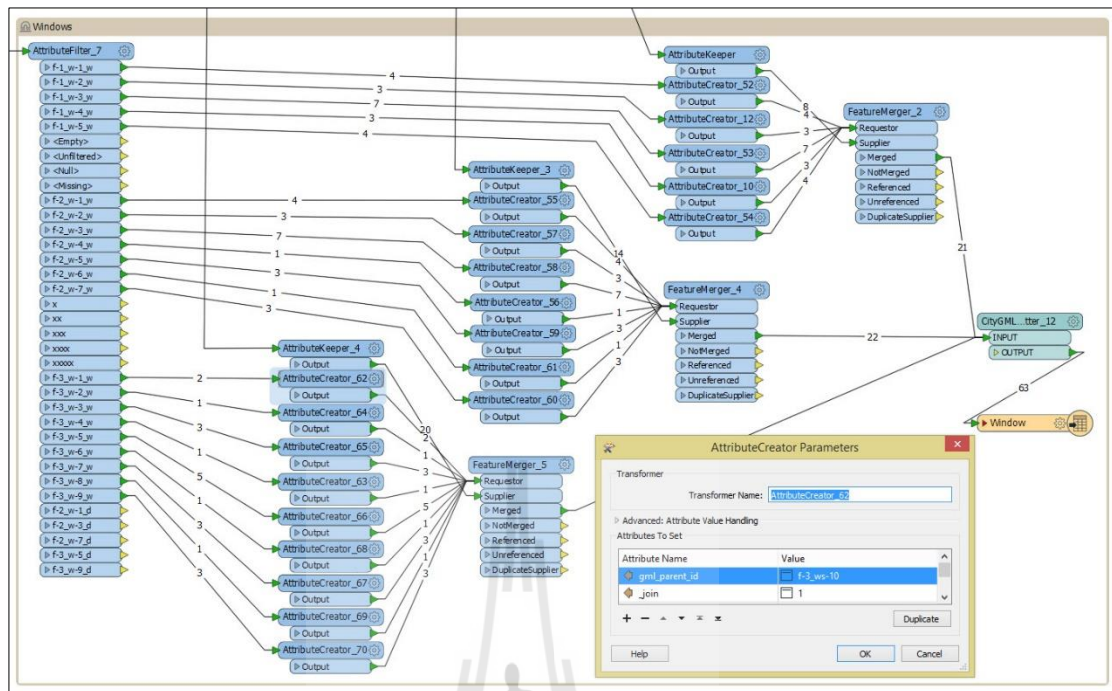


Figure 4.13 Translation of windows.

Basically FME SketchUp reader reads skp file as a single piece, therefore using Deaggregator tool SketchUp model was breakup into its multiple geometries. These geometries were assigned with universal unique identifier generated by the UUIDGenerator tool. FME reads SketchUp layers as traits in the geometry rather than as an attribute. The tool GeometryPropertyExtractor extracts those traits to feature attributes. The extracted attributes need to be exposed to be used by other transformers later, so it was exposed using AttributeExposer tool. Hereafter the individual SketchUp layers were translated to its respective feature classes of CityGML building model.

The feature classes of CityGML form a hierarchy structure, which need to be maintained during the structuring of the translation. The hierarchy tree of a building starts with building at the top bounded by different thematic surfaces. The

parent identifier of thematic surfaces belonging to a building was assigned the same value as the identifier value assigned to that building. Opening features falls at the base of the tree and it is attached to a thematic surface rather than to building. A copy of the thematic surface to which the opening belongs was passed to AttributeCreator transformer with attributes join and parent identifier having value 1 and identifier of the thematic class respectively. The same fake join identifier was created in the opening class to join it to its parent thematic surface. FeatureMerger merges the opening to its parent thematic class maintaining the hierarchy.

The output from this translation is a GML file, which can be stored or exchanged among the user communities. The translation log file showing the summary of features read and written is shown in the Figure 4.14.

Translation Log	

Features Read Summary	

building footprint	1
georeferenced building footprint merged with 3D model	1
=====	
Total Features Read	2

Features Written Summary	

Building	11
BuildingInstallation	8
CeilingSurface	9
ClosureSurface	6
Door	22
FloorSurface	10
GroundSurface	1
InteriorWallSurface	278
RoofSurface	2
Room	10
WallSurface	42
Window	63
=====	
Total Features Written	462

Translation was SUCCESSFUL with 4 warning(s) (462 feature(s) output)	
FME Session Duration: 14.1 seconds. (CPU: 11.5s user, 1.5s system)	
END - ProcessID: 6276, peak process memory usage: 149076 kB, current process memory usage: 138132 kB	
Translation was SUCCESSFUL	

Figure 4.14 FME translation log file.

Figure 4.15 shows the eleven buildings resulted from skp2citygml translation in FME Data Inspector with total number of features created under individual CityGML classes.

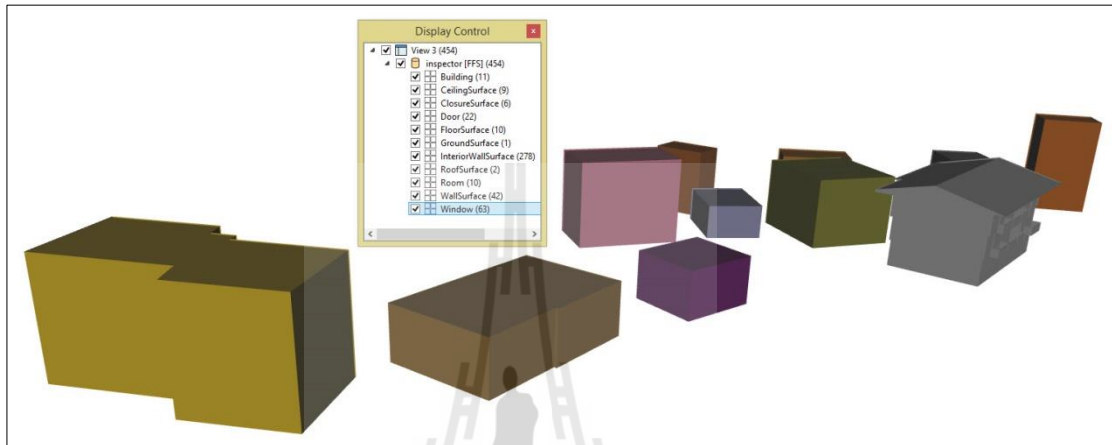


Figure 4.15 Output of the FME translation in FME Data Inspector.

As results it revealed that when the detail of building increases, the workflow of translation in FME becomes complex, especially when we have to maintain hierarchical characteristic of a building. For example room is a part of building and it composes of ceiling, floor, and interior wall. While finalizing this output, many other difficulties were observed too. This study requires designing CityGML ADE which is supposed to hold the legal information of a building and it need to be fed with data in this translation. Though FME provides mechanism to attach ADE for writing the input data, but it does not document the process.

At the initial stage of research, while running the translation, the process was continuously running without any result. This problem was shared to the support team of FME and it was finally solved. The problem occurred due to the presence of two versions of GML in the CityGML ADE. While CityGML 2.0 was

defined based on GML 3.1.1 and ADE was created with GML 3.2.1. This mixture of GML 3.1.1 and 3.2.1 in ADE file is not supported by FME.

4.3 UML model of CityGML ADE for 3D Cadastre

The new properties defined in section 4.1 were integrated to the CityGML Building model. These classes provide the reference to understand the location and size of the legal objects. It represents features which defines the characteristics and properties of 3D cadastre for building.

Figure 4.16 shows the UML class diagram of CityGML ADE for 3D Cadastre. The classes of CityGML and the new classes of ADE were highlighted as are colored yellow and pink. Codelists were defined for the class common property and accessory unit as well as for the attribute floor number (shown as blue color in Figure 4.5). Codelist definition provides a mechanism to defines some predefined attributes for classes based on NLCS requirement.

The classes defining the legal information of a building were added as subclass of abstract class `LegalSpaceBuildingUnit`, which is itself a subclass of abstract superclass `SpatialUnit`. These abstract classes from the ISO standard LADM were defined as subclass to the abstract root class `CityObject` of CityGML. The abstract class `SpatialUnit` was defined here to extend the scope of all the 3D legal properties such as building, utilities, tunnels, hydropower, etc. However, in this study only the building part was introduced, therefore, the abstract class `LegalSpaceBuildingUnit` was included to cater all the aspect of 3D legal properties defined by a building. The other 3D properties can be added later as subclasses of `SpatialUnit` when required.

As shown in the UML diagram, this ADE does not define geometry for 3D legal objects; rather it provides legal information and a link to the geometry. The geometry was stored in the class room and building installation. Since class room definition was not needed in this work, it was used to store the geometry of apartment unit and common property unit and thereby reducing redundant geometry. Moreover, FME does not provide documentation on citygml-feature-role definition which is necessary to write geometries to CityGML ADE classes.

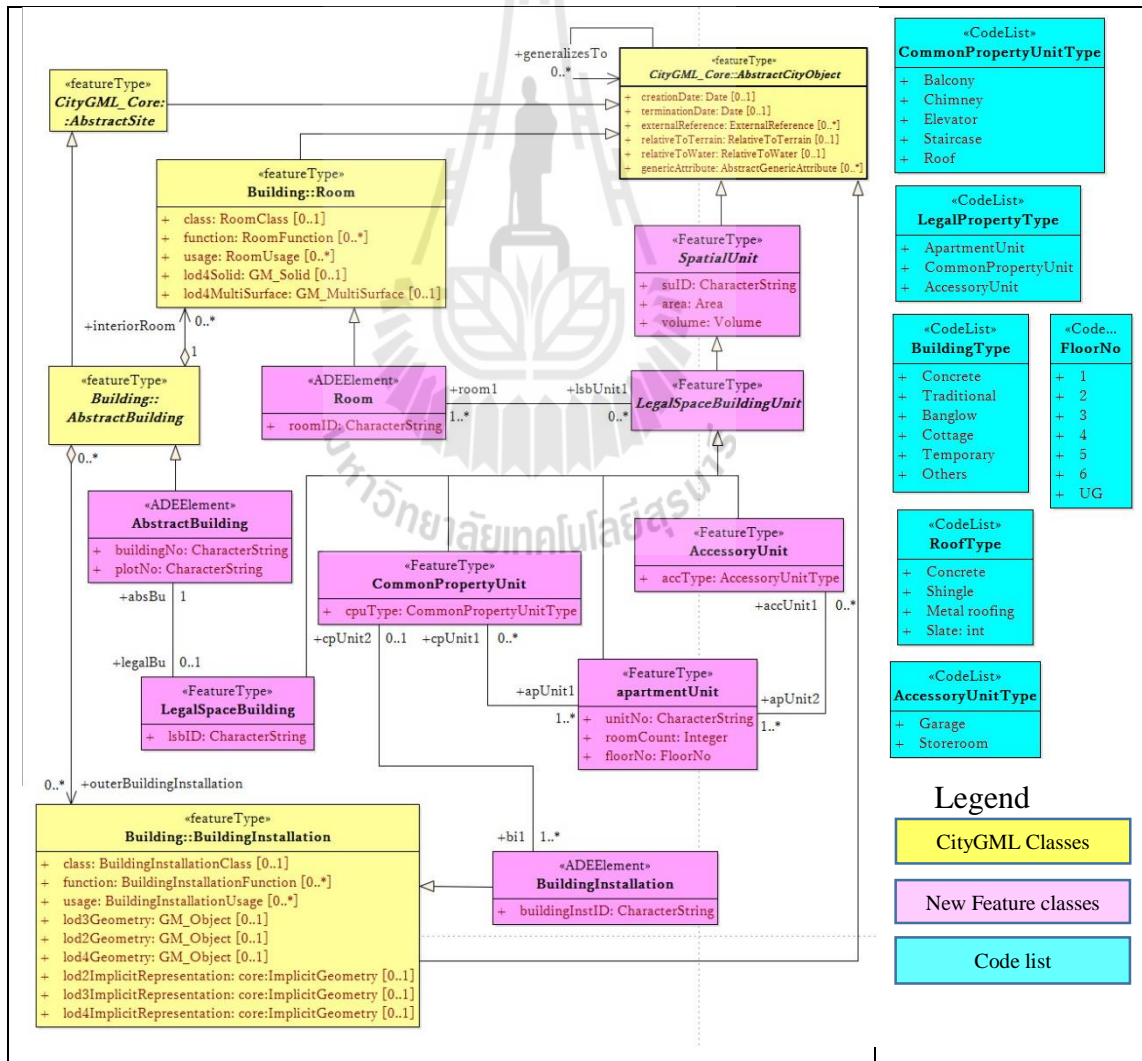


Figure 4.16 UML diagram of Cadastre ADE.

4.4 Data exchange format development

ShapeChange software was used to generate XML schema from the UML model of Cadastre ADE. It is an open-source Java tool for generating schemas and associated resources from a UML domain model. This tool reads XML Interchange Files (XMI 1.0) or Enterprise Architect Project (EAP) as an input model and creates one or more target representations such as XML Schema, JSON, RDF, KML, codelist dictionaries, Schematron documents, and Feature catalogues. Figure 4.17 illustrates the process of ShapeChange.

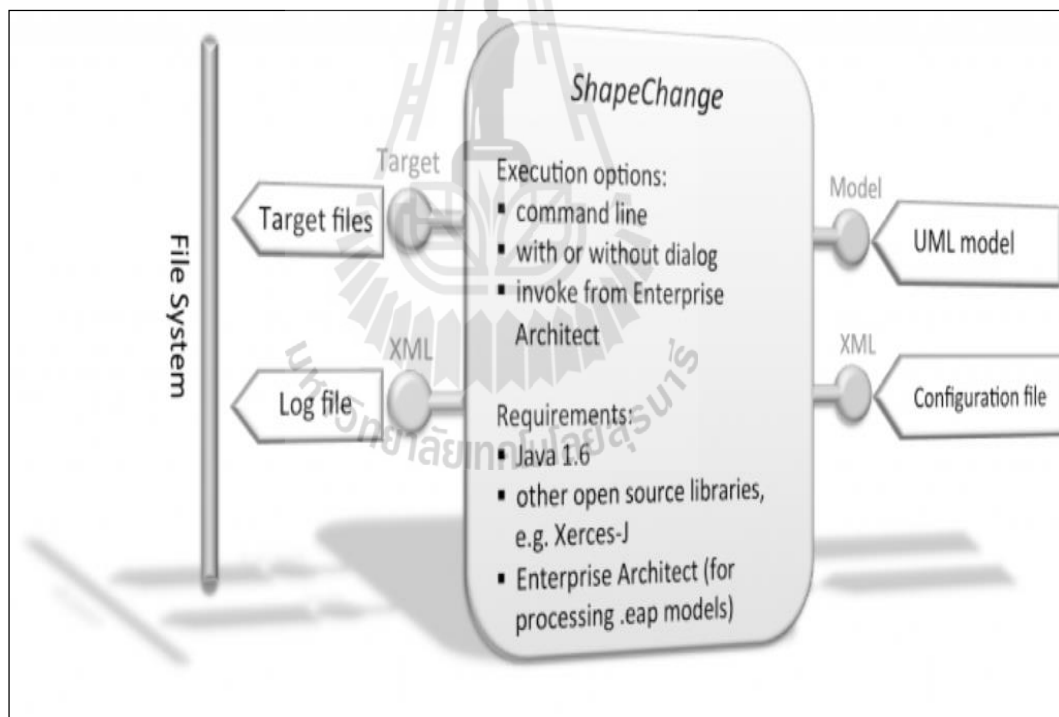


Figure 4.17 Process of ShapeChange (shapechange, 2015).

The required input and output parameters are set in a configuration file. A configuration file is an XML file, which conforms to a custom XML Schema (ShapeChangeConfiguration.xsd). It forms the primary mechanism for providing

arguments to ShapeChange. The configuration file as shown in Figure 4.18 consists of three functional elements namely:

(1) **<input> element** - Specify the location and format (xmi/eap) of the source UML model; which application schema packages are to be converted.

(2) **<log> element** - Specify the name and destination location of the log file and the logging level.

(3) **<targets> element** - Specify the destination locations, formats, encoding rules and any additional mapping rules for each of the required outputs.

Based on the literature available in website (<http://wiki.ieee-earth.org>) entitled data modeling needs & GML, the configuration file was setup for generating XML schema definition in GML version 3.1.1 matching the definition of CityGML version 2.0.0 used here.

Here the UML model was input to ShapeChange as EAP format as suggested by Clemens Portele (personnel communication). ShapeChange tool was activated through a batch file customized in EA and resulted in the log file and the target files (Figure 4.19). The **<targets>** element was set to XML schema and codelist dictionary: CadastreADE.xsd, AccessoryUnitType.xml, LegalPropertyType.xml, FloorNo.xml, CommonPropertyUnitType.xml, RoofType.xml, and BuildingType.xml.

The generated XML Schema and codelists is shown in Appendix A.

```

<?xml version="1.0" encoding="UTF-8"?>
<ShapeChangeConfiguration xmlns:xi="http://www.w3.org/2001/XInclude" xmlns="http://www.interactive-instruments.de/ShapeChange/Configuration/1.1" xmlns:sc="
http://www.interactive-instruments.de/ShapeChange/Configuration/1.1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
http://www.interactive-instruments.de/ShapeChange/Configuration/1.1 http://shapechange.net/resources/schema/ShapeChangeConfiguration.xsd">
  <input>
    <!--Input Parameter-->
    <parameter name="inputModelType" value="EA7"/>
    <parameter name="inputFile" value="E:\Master Programme\4. Fourth Term\Research Documents\Research Work\Analysis\UML Modelling\CityGML_ADE.eap"/>
    <parameter name="appSchemaName" value="CadastraADE"/>
    <parameter name="publicOnly" value="true"/>
    <parameter name="checkingConstraints" value="disabled"/>
    <parameter name="sortedOutput" value="true"/>
    <parameter name="addTaggedValues" value=""/>
    <xi:include href="http://shapechange.net/resources/config/StandardAliases.xml"/>
  </input>
  <log>
    <parameter name="reportLevel" value="INFO"/>
    <parameter name="logFile" value="result.xml"/>
  </log>
  <targets>
    <!--Output Parameter-->
    <TargetXmlSchema class="de.interactive_instruments.ShapeChange.Target.XmlSchema.XmlSchema" mode="enabled">
      <targetParameter name="outputDirectory" value="C:/ShapeChange/output/CadastraADESchema"/>
      <targetParameter name="sortedOutput" value="true"/>
      <targetParameter name="defaultEncodingRule" value="citygml-ade"/>
      <xi:include href="http://shapechange.net/resources/config/StandardRules.xml"/>
      <xi:include href="C:/ShapeChange/test/oonfig/StandardNamespacesGML311.xml"/>
      <xi:include href="C:/ShapeChange/test/config/StandardMapEntries-v31.xml"/>
      <xmlNamespaces>
        <XmlNamespace nsabr="oore" ns="http://www.opengis.net/citygml/2.0" location="http://schemas.opengis.net/citygml/2.0/cityGMLBase.xsd"/>
        <XmlNamespace nsabr="bldg" ns="http://www.opengis.net/citygml/building/2.0" location="http://schemas.opengis.net/citygml/building/2.0/building.xsd"/>
      </xmlNamespaces>
    </TargetXmlSchema>
    <!-- Codelist Dictionaries-->
    <Target class="de.interactive_instruments.ShapeChange.Target.Codellists.CodelistDictionaries" mode="enabled">
      <targetParameter name="outputDirectory" value="C:/ShapeChange/output/CadastraADESchema/codelist"/>
      <targetParameter name="outputFilename" value="CadastraADECodelist"/>
      <targetParameter name="sortedOutput" value="true"/>
      <targetParameter name="enumerations" value="false"/>
    </Target>
  </targets>
</ShapeChangeConfiguration>

```

Figure 4.18 Configuration file of ShapeChange in XML format.

ShapeChange result

Result:	Conversion Completed
Start time:	Wed Feb 10 23:29:56 ICT 2016
End time:	Wed Feb 10 23:35:11 ICT 2016
Configuration file:	C:\ShapeChange\ShapeChangeConfiguration.xml
ShapeChange version:	2.0.1

Messages

[Debug](#) | [Info](#) | [Warning](#) | [Error](#) | [Fatal Error](#)

Severity	Message	Source
Info	Stereotype <<adeelement>> of class 'Building' is not an allowed value in encoding rule 'iso19136_2007' and is ignored.	
Info	Stereotype <<adeelement>> of class 'CityFurniture' is not an allowed value in encoding rule 'iso19136_2007' and is ignored.	
Info	Stereotype <<adeelement>> of class 'Railway' is not an allowed value in encoding rule 'iso19136_2007' and is ignored.	
Info	Stereotype <<adeelement>> of class 'Road' is not an allowed value in encoding rule 'iso19136_2007' and is ignored.	
Info	Application schema found, package name: 'CadastrADE', target namespace: 'file:///C:/ShapeChange/output/CadastrADESchema'	
Info	(Converter.java) Now processing target 'INPUT':	
Info	(Converter.java) Executed target class 'de.interactive_instruments.ShapeChange.Target.XmlSchema.XmlSchema' for input ID: 'INPUT':	
Info	Application schema found, package name: 'CadastrADE', target namespace: 'file:///C:/ShapeChange/output/CadastrADESchema'	
Info	(Converter.java) Now processing target 'INPUT':	
Info	(Converter.java) Executed target class 'de.interactive_instruments.ShapeChange.Target.CodeLists.CodeListDictionaries' for input ID: 'INPUT':	

Results

Target	Scope	File
Code List Dictionary	cad:AccessoryUnitType	AccessoryUnitType.xml
Code List Dictionary	cad:BuildingType	BuildingType.xml
Code List Dictionary	cad:CommonPropertyUnitType	CommonPropertyUnitType.xml
Code List Dictionary	cad:FloorNo	FloorNo.xml
Code List Dictionary	cad:LegalPropertyType	LegalPropertyType.xml
Code List Dictionary	cad:RoofType	RoofType.xml
XML Schema	file:///C:/ShapeChange/output/CadastrADESchema	CadastrADE.xsd

Figure 4.19 The log file displaying the results and messages from ShapeChange tool.

4.5 Building database

The 3D building database covers all modeling aspects of building module and core module of CityGML 2.0.0 as well as the CityGML ADE for 3D cadastre. The details of 3D city database schema development in the documentation of “3D City Database” version 3.0.0 was used here to design the 3D building database. 3D City Database is an open source software package available in the website www.3dcity.com. In this release the 3DCityDB was implemented in the commercial

database, Oracle and the Open Source database, PostGIS. However, in this research, the 3D building database schema was realized in Microsoft SQL Server 2008, which is the database system in NLCS.

Generally in database modeling, one or more classes of the UML diagram are mapped onto one table keeping the name identical to the class name. The attributes of the classes become columns in the corresponding table with identical name as well. Data types specified in CityGML were substituted by data types of the selected database allowing an effective representation (see Table 4.2). The data types available for the MS SQL Server 2008 are shown in Appendix B.

Table 4.2 Data type mapping.

UML	MS SQL Server
String, any URI	varchar, char, text
Integer	int
Double, gml:LengthType	float, real
Boolean	bit
Date	date, datetime
Enumeration	varchar
GML Geometry	Geometry

This section is elaborated in the following sub-sections starting with the simplification of UML diagram to the implementation of the database schema into the relational database.

4.5.1 Simplification of CityGML and ADE for 3D Cadastre

CityGML uses concepts which are seen as quite complex by some users and software implementers. Direct translation to relational data model would give inefficient result (Karpina, 2014). Therefore, some simplifications were

considered to simplify the implementation and consequently enhancing the performance of database.

Figure 4.20 and Figure 4.21 show the UML diagram of core model and building model respectively. Classes which were combined into a single table according to the class relationship are shown in the UML diagrams as orange colored boxes. The green ones shows the many-to-many (m:n) relationship between classes, which have to be mapped into a separate table.

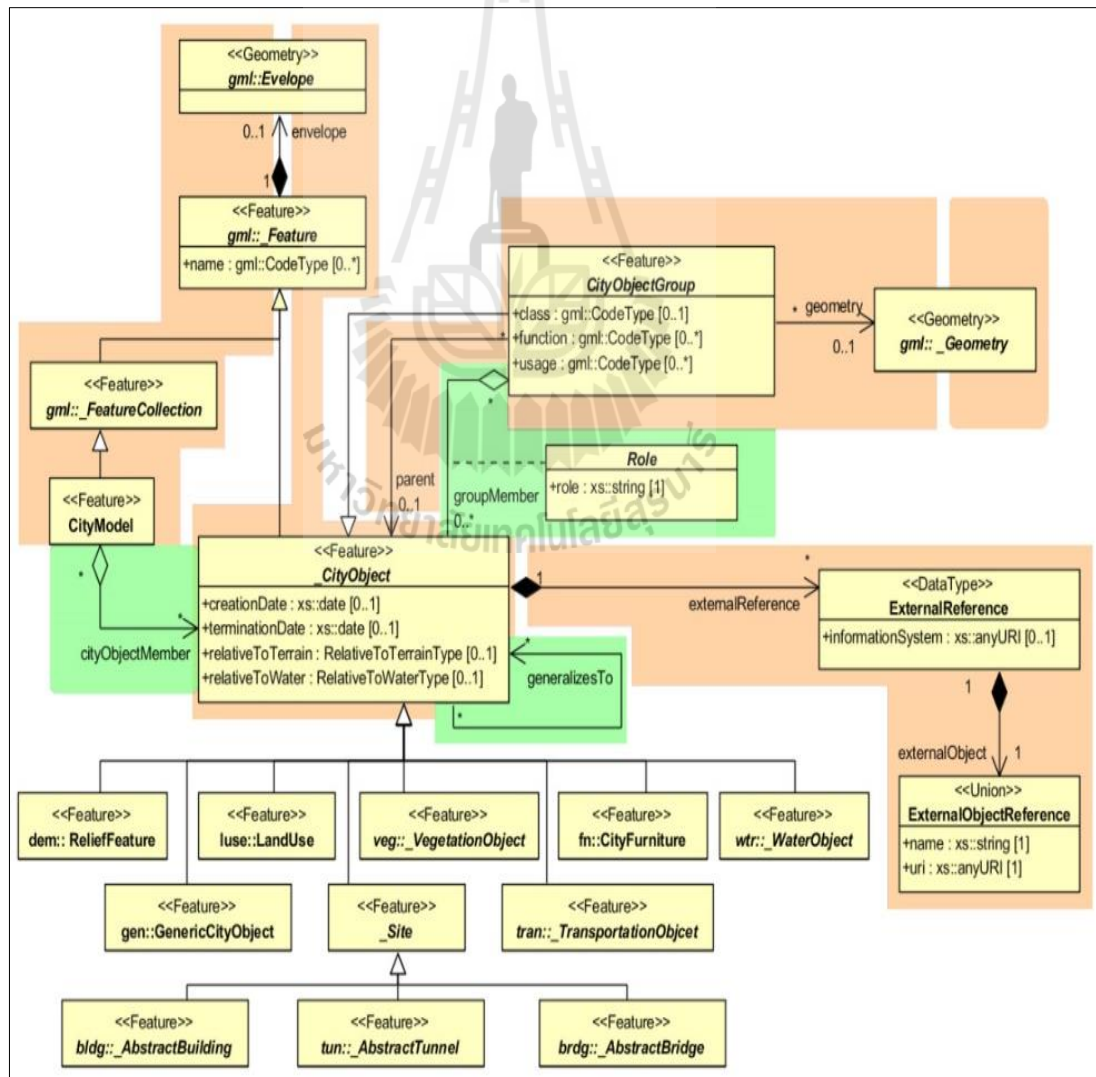


Figure 4.20 UML diagram of core model (3dcitydb, 2015).

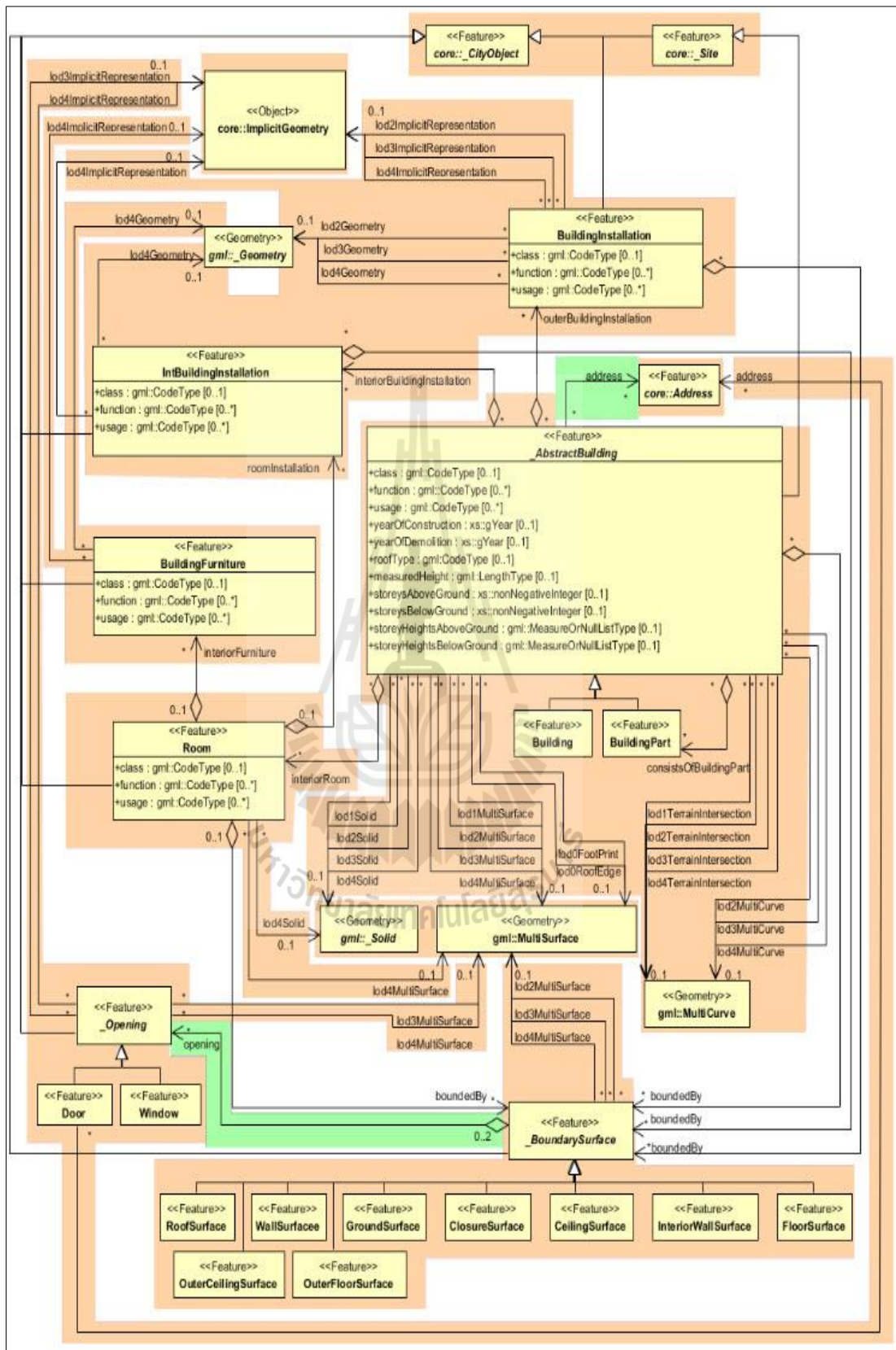


Figure 4.21 UML diagram of building model (3dcitydb, 2015).

Meanwhile, Figure 4.22 and 4.23 show the simplified UML diagram of core and building model of CityGML 2.0.0 which was used to generate tables of database schema. Based on Figure 4.21, the following simplification was done to the building model:

(1) Class `BuildingInstallation` and `IntBuildingInstallation` was combined together as a single class `BuildingInstallation` to store both the exterior and interior features attached to a building.

(2) The three CityGML classes `AbstractBuilding`, `Building` and `BuildingPart` are merged into a single table `Building` with all the geometries added as attributes.

(3) Class `BoundarySurface` and its subclasses was defined to a single class called `ThematicSurface`.

(4) Class `Opening` and its subclass `Door` and `Window` was remodel as class `Opening`.



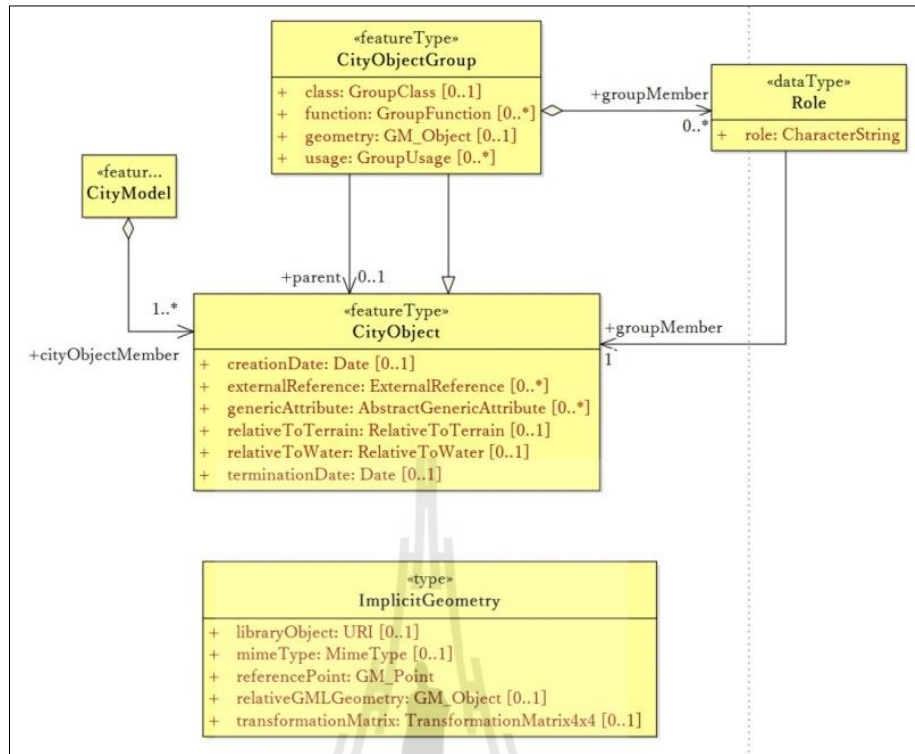


Figure 4.22 The simplified UML diagram of Core model.

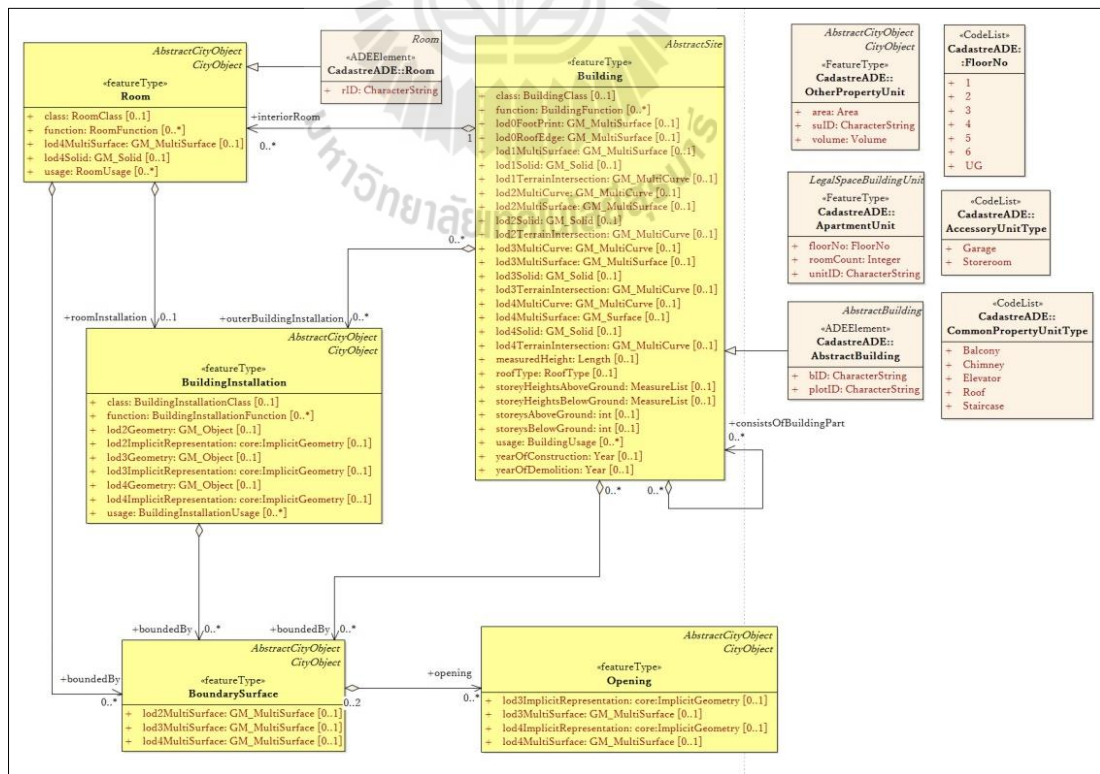


Figure 4.23 The simplified UML diagram of building model and ADE.

4.5.2 Database modeling

The modeling of the 3D building database was done in EA, which supports comprehensive functionality for modeling database structures. UML diagram of tables were generated from the simplified UML diagram of core model, building model and ADE automatically using the tool MDA transformation of EA. The resulted tables were edited in EA to correct the name of tables and columns as well as addition of new columns and relationships to design a comprehensive database schema. The code lists defined in the CityGML ADE were translated into tables with a unique identifier column, name and description of the code list. The database schema was described and displayed graphically in the following sections.

4.5.2.1 Core model

Figure 4.24 displays the tables of database schema of the core model included ObjectClass, CityObject, CityModel, and CityObjectGroup. The characteristics of each table were separately explained as below.

(1) ObjectClass: Table ObjectClass manages all class names of the schema (see Table 4.3). It efficiently determines the affiliation to a class in the superclass tables. The relation of the subclass to its parent class is represented via the attribute superClassID in the subclass as a foreign key (FK) to the ID of the parent class.

(2) CityObject: All city objects are represented by tuples in the table CityObject. The fields are identical to the attributes of the corresponding UML class, plus additional columns for metadata like lastModificationDate, updatingPerson, reasonForUpdate and lineage. The BoundingBox (envelope) was realized as rectangular geometry using five points, that join the minimum and

Table 4.3 Definition of Class names.

ID	className	superClassID
0	Undefined	
1	_GML	
2	_Feature	1
3	_CityObject	2
4	_Site	3
5	CityObjectGroup	3
6	_AbstractBuilding	4
7	BuildingPart	6
8	Building	6
9	BuildingInstallation	3
10	_BuildingBoundarySurface	3
11	BuildingCeilingSurface	10
12	InteriorBuildingWallSurface	10
13	BuildingFloorSurface	10
14	BuildingRoofSurface	10
15	BuildingWallSurface	10
16	BuildingGroundSurface	10
17	BuildingClosureSurface	10
18	_BuildingOpening	3
19	BuildingWindow	18
20	BuildingDoor	18
21	BuildingRoom	3
22	FeatureCollection	2
23	CityModel	22
24	OuterBuildingCeilingSurface	10
25	OuterBuildingFloorSurface	10
26	_SpatialUnit	3
27	_LegalSpaceBuildingUnit	26
28	ApartmentUnit	27
29	CommonPropertyUnit	27
30	AccessoryUnit	27
31	LegalSpaceBuilding	27

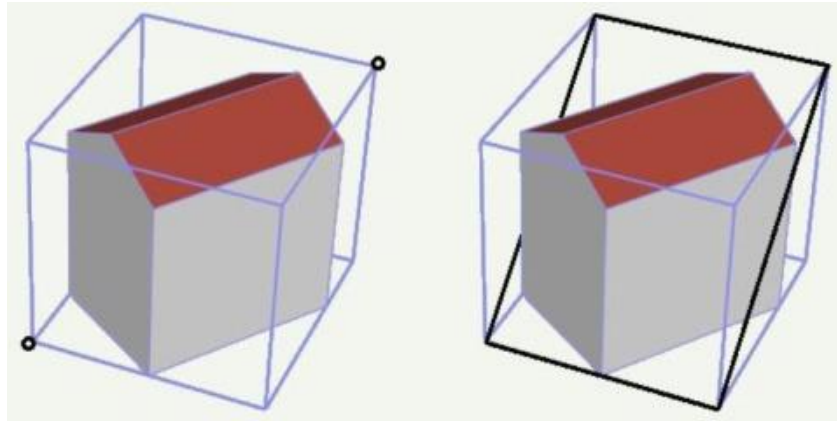


Figure 4.25 CityObject envelope storage as a 3D rectangle specified by (left: the two black points with minimum and maximum coordinate values respectively) and (right: black polygon using five points) (3dcitydb, 2015).

4.5.2.2 Building model and ADE

The building and ADE model is realized by the tables shown in Figure 4.26 included building, thematic surface, building installation, opening, and apartment.

(1) Building: The hierarchy within a building is realized by the FK buildingParentID which refers to the superordinate building and contains NULL, if such does not exist. This way, a tree-like structure arises for building aggregates. FK buildingRootID refers directly to the top level (root) of a building tree. This FK provides a mean to select all parts in a building tree by selecting those with the same value. There are two columns defined in the table to specify the type of roof and building. A unit column is provided for every attribute including measure information like measuredHeight or storeyHeightsAboveGround etc. to specify the scale. Geometry is represented by several FKs lod0FootPrintID, lod0RoofPrintID, lodxMultiSurfaceID (x = 1 and 4), and lodxSolidID (x = 1 and 4) which refer to entries in the SurfaceGeometry table.

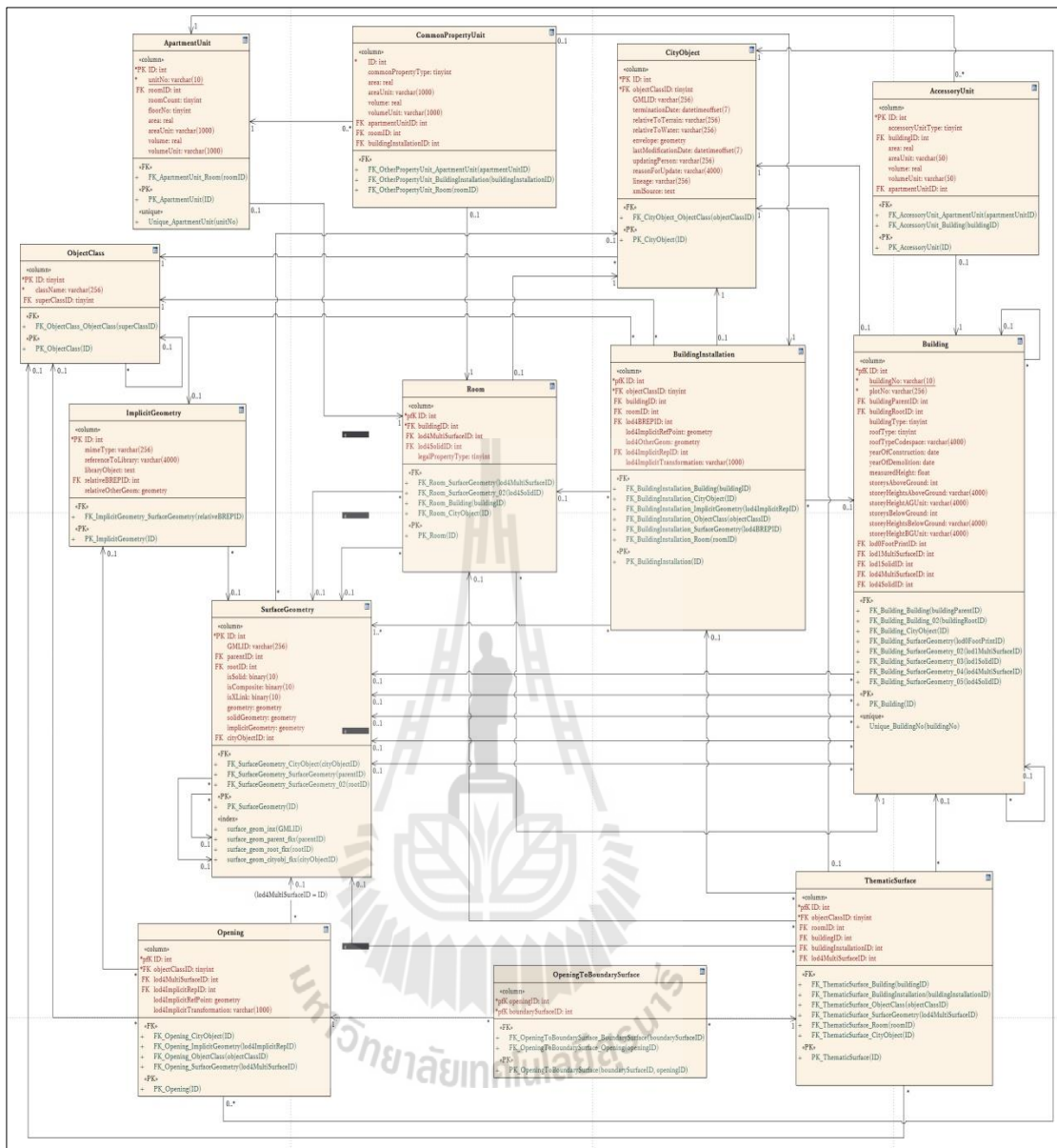


Figure 4.26 Building database schema.

(2) **ThematicSurface**: The table ThematicSurface represents all different types of thematic boundary features represented by class BoundarySurface and its subclasses. The type of the boundary surface was given by the FK objectClassID, which link to the table ObjectClass. Allowed integer values of objectClassID in the ThematicSurface table are shown in Table 4.4.

Table 4.4 Allowed integer values of objectClassID in the ThematicSurface table.

objectClassID	className
11	BuildingCeilingSurface
12	InteriorBuildingWallSurface
13	BuildingFloorSurface
14	BuildingRoofSurface
15	BuildingWallSurface
16	BuildingGroundSurface
17	BuildingClosureSurface
24	OuterBuildingCeilingSurface
25	OuterBuildingFloorSurface

The aggregation relation between buildings and the corresponding boundary surfaces results from the FK buildingID which refers to the ID of the respective building. Thematic surfaces and the corresponding parent feature share their geometry whereby the geometry was defined only once and used conjointly as XLinks. The SurfaceGeometry, which for example geometrically defines a roof, should at the same time be a part of the volume geometry of the parent feature the roof belongs to.

(3) **BuildingInstallation:** The UML classes BuildingInstallation and IntBuildingInstallation were realized by the single table BuildingInstallation. The attribute objectClassID (external 27, internal 28) distinguishes internal and external objects attached to a building. FK buildingID or roomID maintains the relation to the corresponding parent feature, whereas the surface based geometry was given via the FK lod4BREPID referring to the table SurfaceGeometry. CityGML 2.0.0 building installations can also be represented by using prototypes which are stored as library objects implicitly. The information needed for mapping prototype objects to buildings consists of base point geometry (lod4ImplicitRefPoint), a transformation matrix (lod4ImplicitTransformation), which

is stored as a string, and a FK reference to the ImplicitGeometry table (lod4ImplicitRepID) where a reference to an explicit surface based geometry is saved.

(4) Opening: Table Opening represents CityGML class Opening and its subclasses Window and Door. The differentiation was achieved by the FK objectClassID which refers to the attribute ID of the table ObjectClass. Valid integer values of objectClassID are '20' (Door) and '19' (Window). Table OpeningToThemSurface associates an opening ID in table Opening with a thematic surface ID in table ThematicSurface representing the m:n relation between both tables. Similar to building installations openings consists of columns lod4ImplicitRepID, lod4ImplicitRefPoint, and lod4ImplicitTransformation model via implicit geometry.

(5) Room: Room objects are allowed in LoD4 only. Therefore the only keys lod4MultiSurfaceID and lod4SolidID refers to the table SurfaceGeometry. Additionally the FKs to tables Building and CityObject are necessary to map the relationship to these tables. In this research, a separate geometry was not defined for the subclasses of LegalSpaceBuilding class. Instead it links to the geometry of room, which is constructed to define an apartment or common property unit within a building.

(6) Individual table was derived for classes: ApartmentUnit, AccessoryUnit, and CommonPropertyUnit. ApartmentUnit consist FK roomID which points to ID in table Room. Similarly, FK roomID and buildingInstallationID of table CommonPropertyUnit links to a tuple in table Room and BuildingInstallation respectively, and FK buildingID in AccessoryUnit refers to a tuple in table Building.

The last two tables have a FK apartmentUnitID which associate to a tuple in apartment unit table.

4.5.2.3 Tables for geometry representation

In the database schema the geometry consists of planar surfaces which correspond each to one entry in the table SurfaceGeometry. The surface-based geometry and implicit geometry was stored by attribute geometry and implicitGeometry respectively. Whereas the volumetric geometry was stored by attribute solidGeometry and its boundary surfaces (outer shell) by attribute geometry as well. A solid is the basis for 3-dimensional geometry. The extent of a solid is defined by the boundary surfaces (shell). A shell is represented by a composite surface, where every shell is used to represent a single connected component of the boundary of a solid.

Figure 4.27 displays geometry hierarchy with the solid geometry at the top and surfaces at the bottom. Surfaces aggregates to form a complex surfaces or the boundary of a volumetric object. The aggregation of multiple surfaces (IDs 6 to 10 in Figure) is realized the way that the newly created surface tuple (ID 2) is not assigned geometry. Instead, the parentID of the surfaces IDs 6 to 10 refer to the ID of ID 2. In addition, a further tuple (ID=1) is introduced, which represent the solid and defines the root element of the whole aggregation structure. Each surface references to its root, using the rootID attribute. This information has big influence on the system performance, as it avoids recursive queries. On the downside there also follows the limitation that each tuple in SurfaceGeometry can only belong to one aggregate.

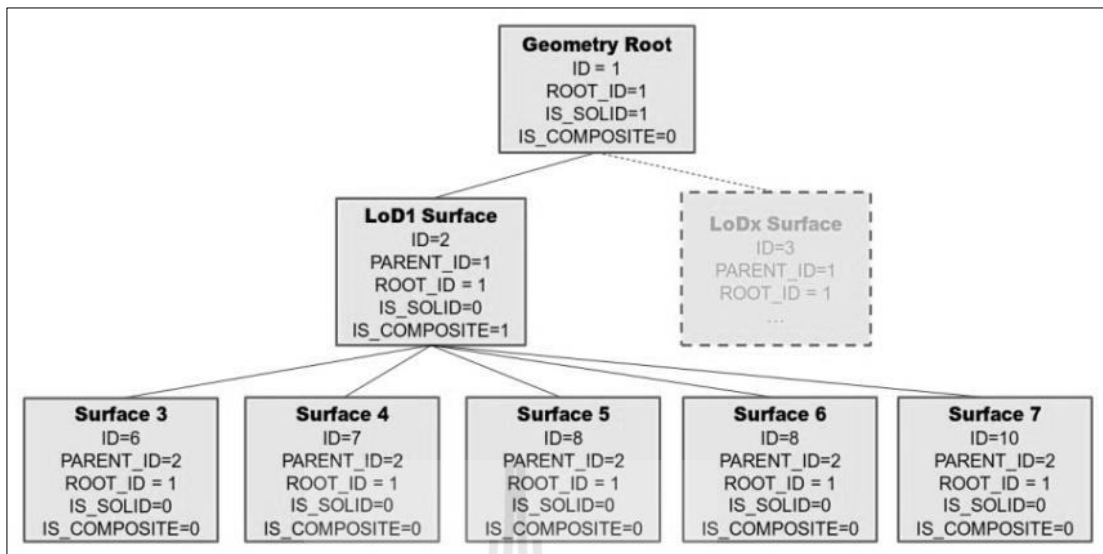


Figure 4.27 Geometry hierarchy (3dcitydb, 2015).

Various flags characterize the type of aggregation: `isSolid` distinguishes between surface (0) and solid (1), and `isComposite` defines whether this is an aggregate (e.g. `MultiSolid`, `MultiSurface`) or a composite (e.g., `CompositeSolid`, `CompositeSurface`) as shown in Table 4.5. Based on these flags the geometry types listed in 5 can be distinguished. To distinguish a `MultiSolid` from a `MultiSurface` its child elements have to be analyzed: In case the child is a `Solid`, the geometry can be identified as `MultiSolid`.

Table 4.5 Attributes determining aggregation types (3dcitydb, 2015).

	<code>isSolid</code>	<code>isComposite</code>	<code>isTriangulated</code>	<code>Geometry</code>	<code>Solid_Geometry</code>
Polygon, Triangle, Rectangle				geometry	NULL
MultiSurface				NULL	NULL
CompositeSurface		*		NULL	NULL
Solid	*			NULL	geometry
MultiSolid				NULL	NULL
CompositeSolid	*	*		NULL	geometry

Aggregated surfaces can be grouped again with other (compound) surfaces, by generating a common parent. This way, arbitrary aggregations of Surfaces, CompositeSurfaces, Solids, CompositeSolids can be formed. Since all tuples in an aggregated geometry refer to the same rootID, all tuples can be retrieved efficiently from the table by selecting those tuples with the same rootID. The aggregation schema allows for the definition of nested aggregations (hierarchy of components). For example, a building geometry (CompositeSolid) can be composed of the house geometry (CompositeSolid) and the garage geometry (Solid), while the house's geometry is further decomposed into the roof geometry (Solid) and the geometry of the house body (Solid). In addition, the FK cityObjectID refers directly to the CityGML features to which the geometry belongs. In order to select all geometries forming the city object one only has to select those with the same cityObjectID.

4.5.2.3 Tables for codelist

Figure 4.28 displays the table of codelists. Each table consists of PK ID, which provides a name and description to a tuple in the table defining the code.

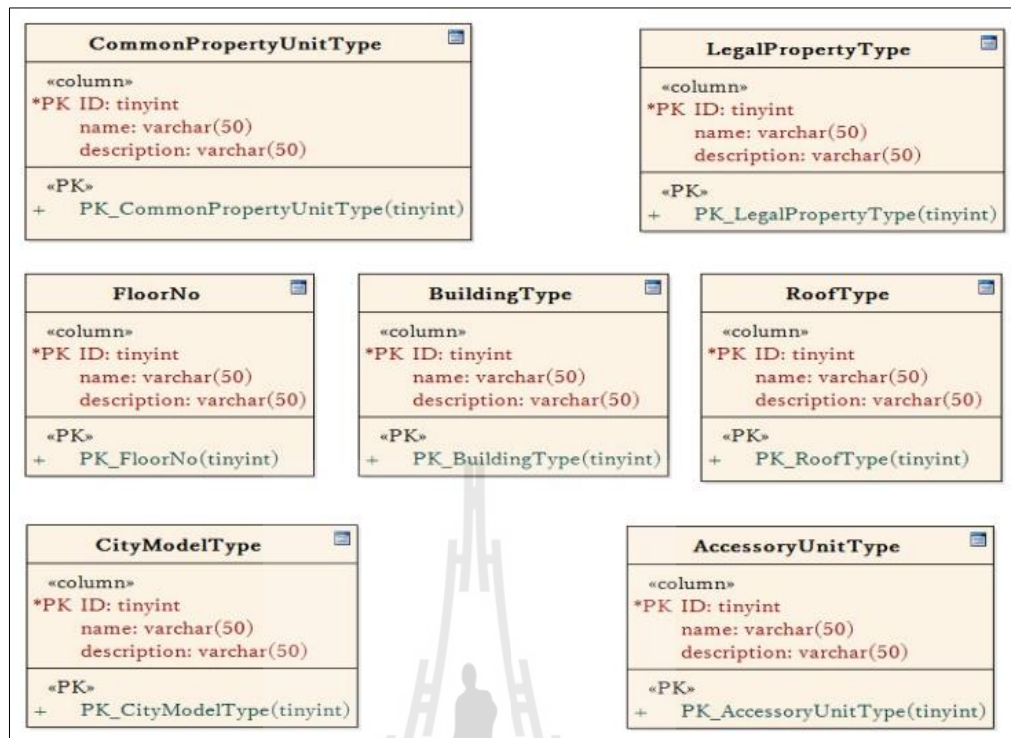


Figure 4.28 Tables of codelist.

4.5.3 Implementation of 3D building cadastral database

Data definition code (DDL) was generated for the each objects of database schema in EA. These DDL scripts can be executed either using the SQL Command Line in EA or directly in Microsoft SQL Server Management Studio. An empty database was created in the Microsoft SQL Server 2008 and assigned name as BuildingDB. Here the generated scripts were executed in the empty database using the first method to create tables, columns, Primary Keys (PK) and FKs, and relationship among the tables.

Additional SQL script was written to generate unique apartment number for the column unitNo in table ApartmentUnit. This unique number was assigned an alphanumeric letter UNIT-X, where X is a running number and it provides a link to its owner maintained in another database. Figure 4.29 displays the

list of tables created in the database based on the database schema diagrams shown in Figure 4.24 and 4.26. Also, tables, column, keys, and relationship of BuildingDB database is graphically displayed in Figure 4.30. All the SQL scripts were presented in Appendix C.

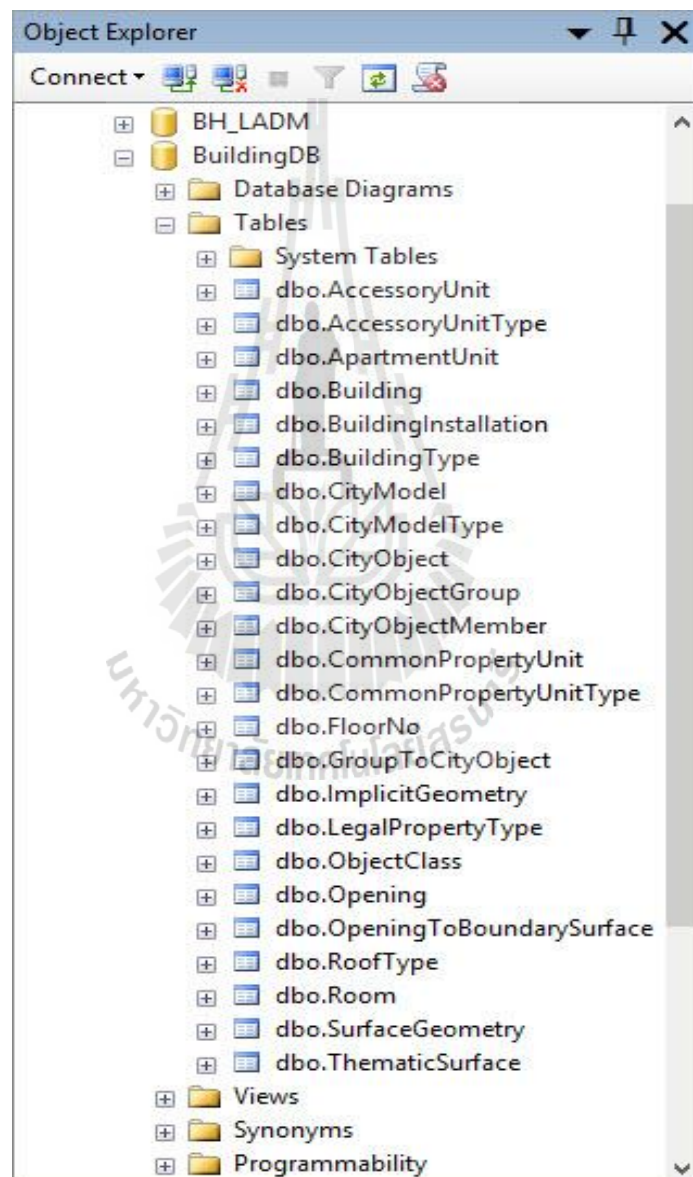


Figure 4.29 List of tables under BuildingDB database in MS SQL Server.

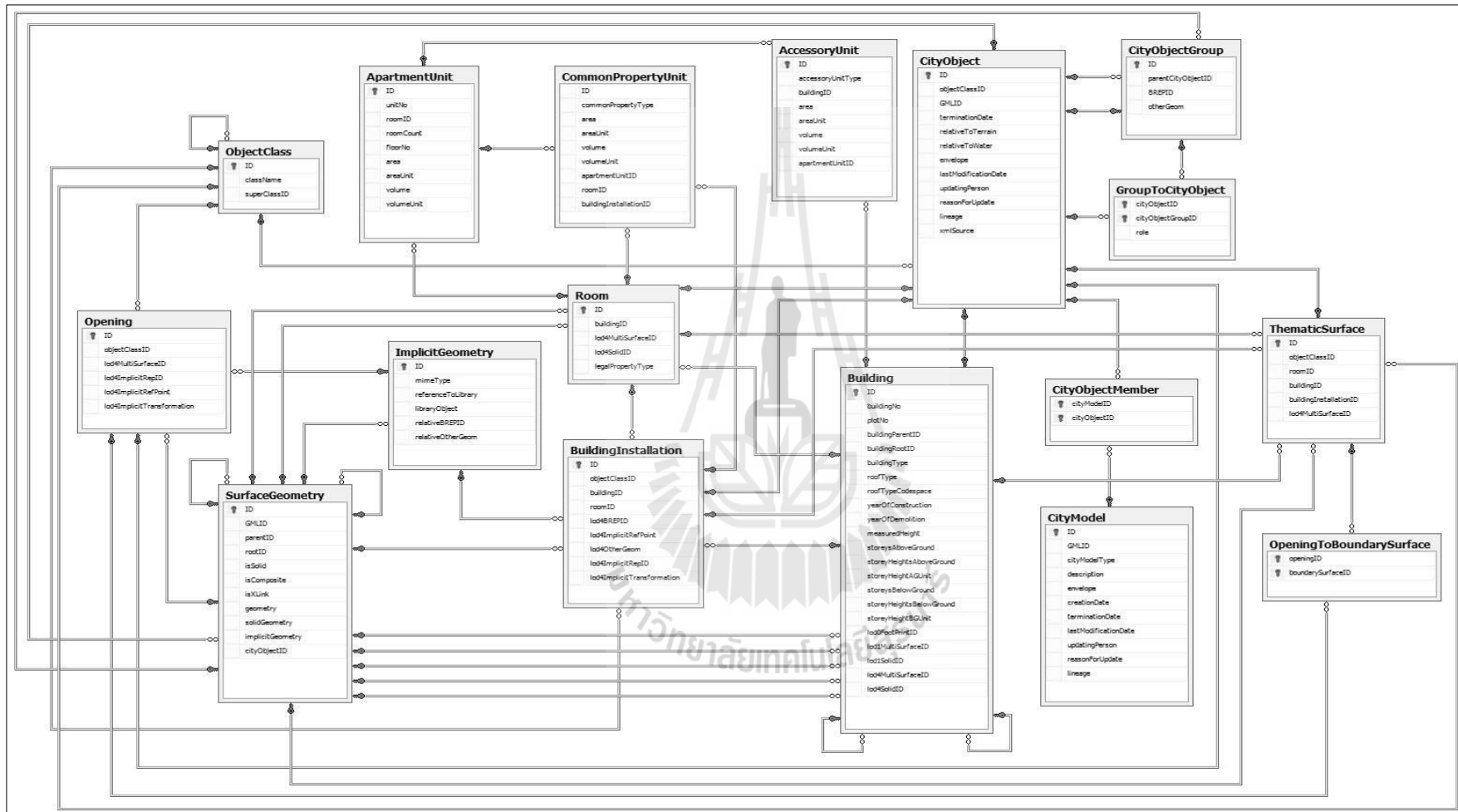


Figure 4.30 Diagram of BuildingDB database (tables, column, keys, and relationship).

4.6 Testing

This section involves two kinds of testing: validation of XML schema and 3D Building model verification. In the final testing phase, the XML schema generated for the 3D cadastre from ShapeChange was validated in XMLSpy software to ensure the validity of the XSD document. The result showed that the syntax of generated XML schema complies with ISO standard and no errors exist as shown in Figure 4.31.

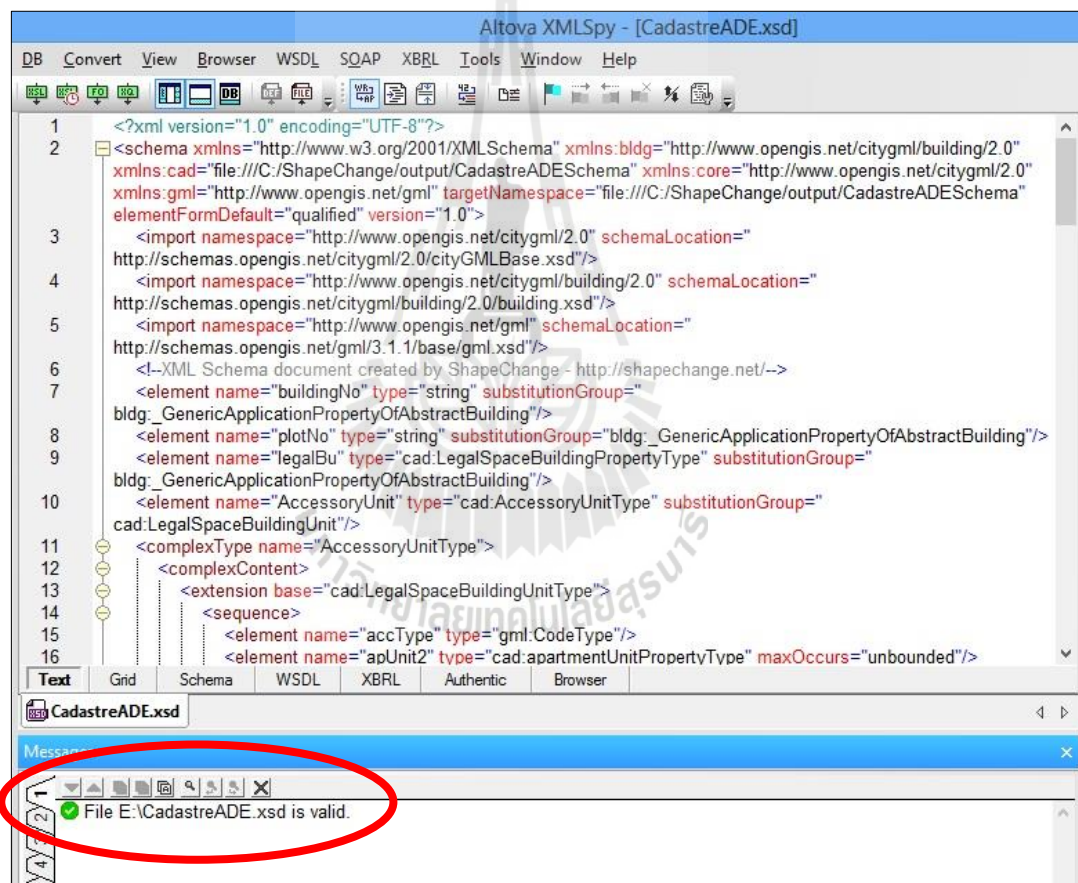


Figure 4.31 XML schema validation result.

In the second test, CityGML instance data resulted from the FME translation was visualized in FZK viewer to verify the expressiveness and correctness of the data structure. Figures 4.32-4.35 show some selected results for expressiveness and

correctness of the data structure of 3D Building model by visualization in the FZK viewer. As shown in Figure 4.32, top view of the 10 LOD1 (blue) and 1 LOD4 (red) buildings were displayed here. This verifies the correct translation of buildings showing the exact number of buildings in the output GML file as in the input SketchUp files. Figure 4.33 displays hierarchical structure of building in the output GML file as a list whereas it is diagrammatically shown in Figure 4.34 and 4.35. These show the correct translation from the input file and display complete list of data structure as hierarchical structure correctly.

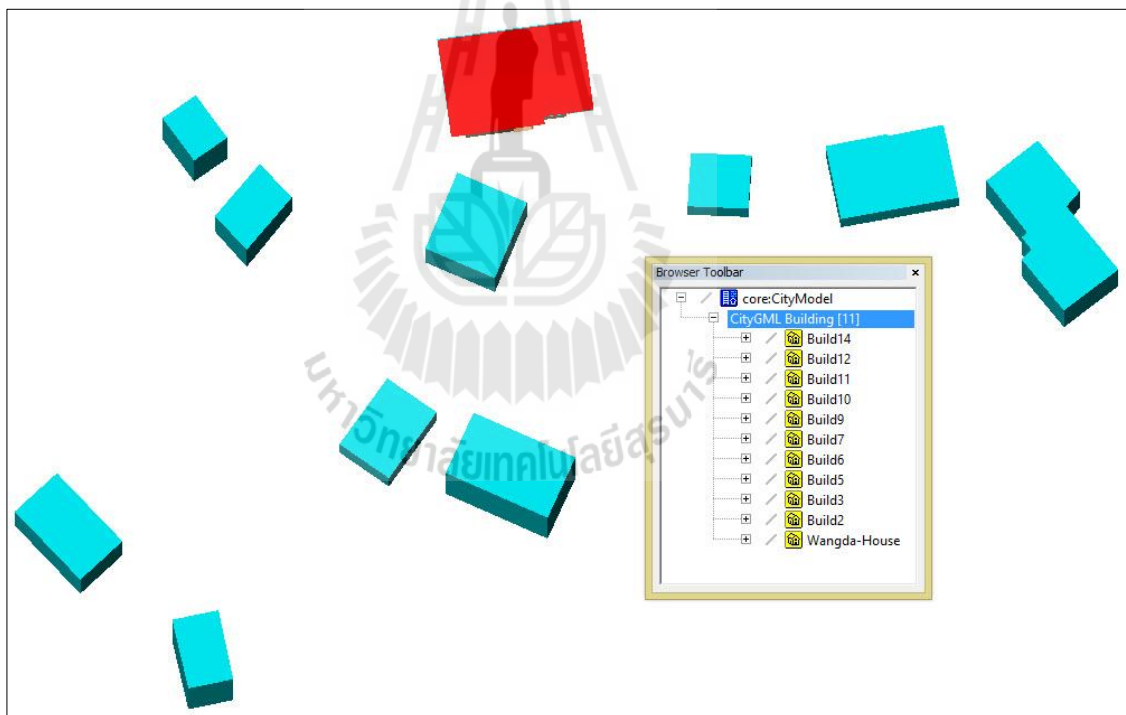


Figure 4.32 Top view and [inset] list of buildings.

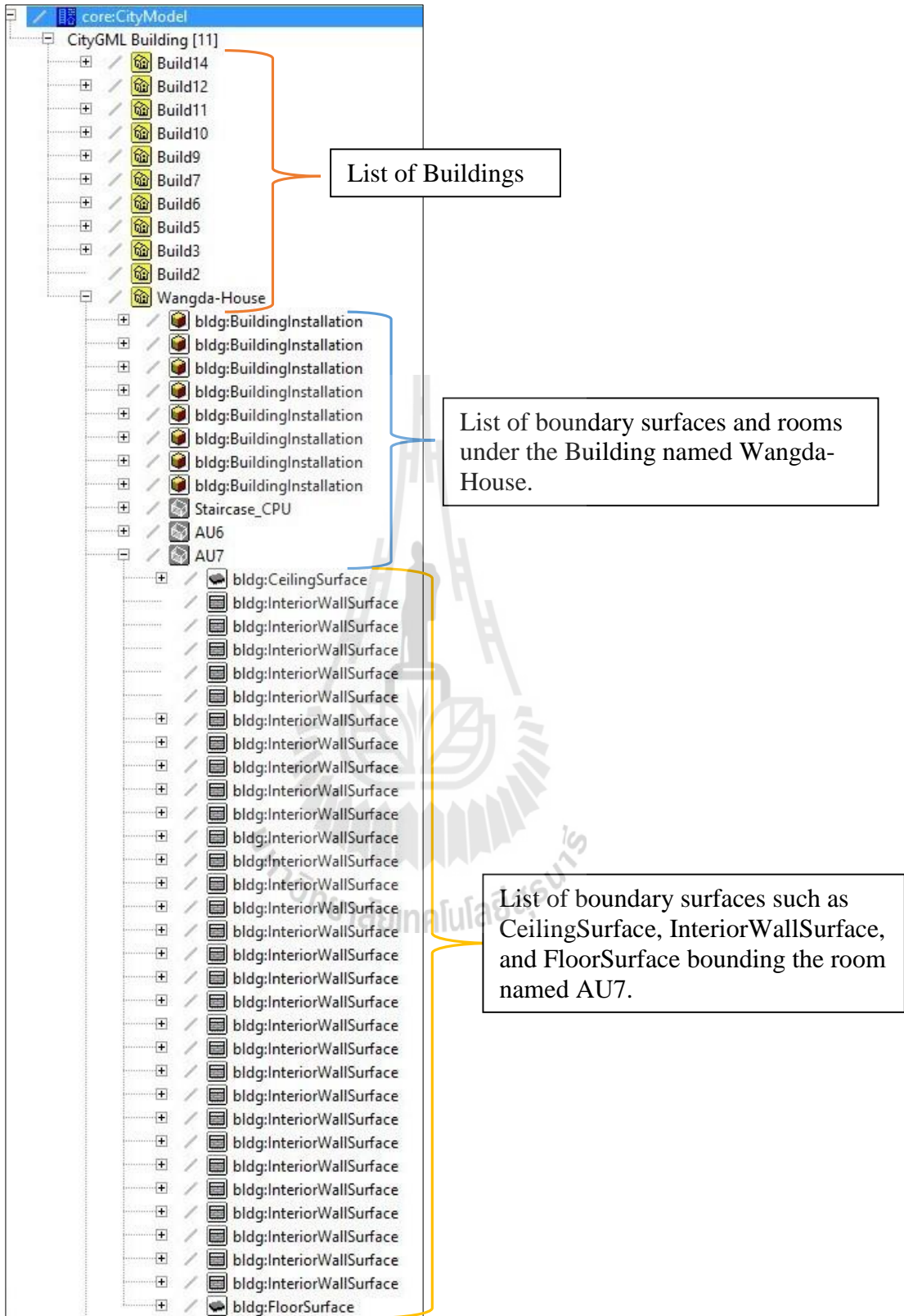


Figure 4.33 A clip from output GML file displaying list of hierarchical ordering of building components.

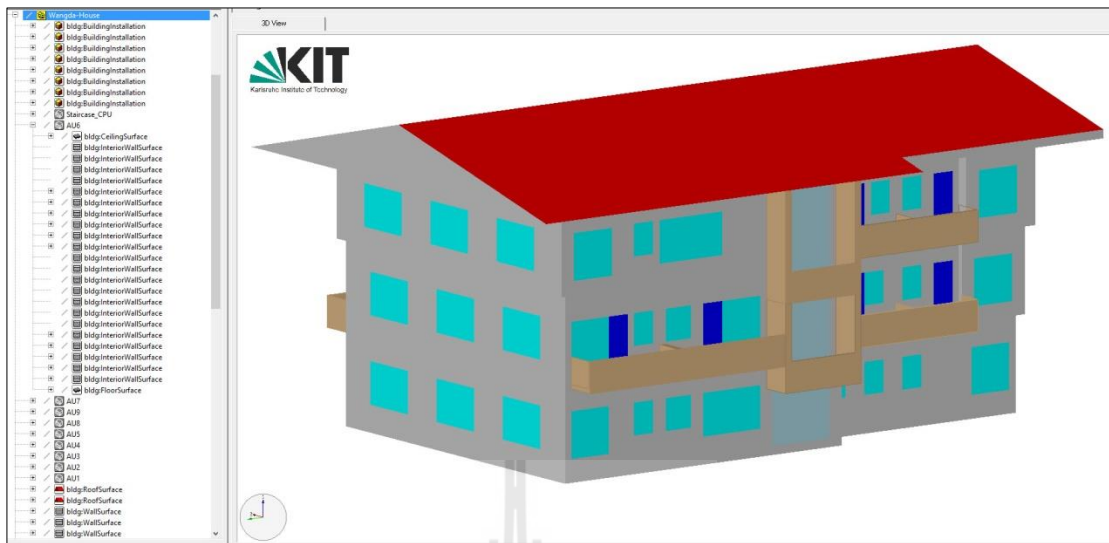


Figure 4.34 Building “Wangda-House” and its components.

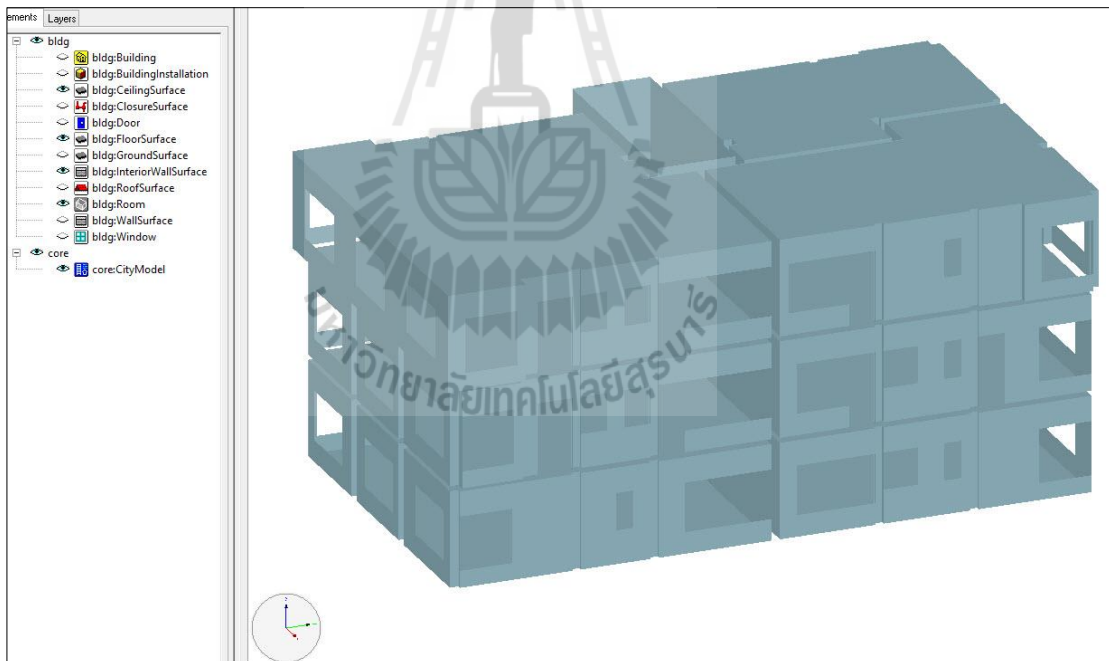


Figure 4.35 Rooms bounded by interior wall surface, ceiling surface, and floor surface.

CHAPTER V

CONCLUSIONS AND RECOMMENDATIONS

This chapter concludes the results of this study based on the defined objectives and give some recommendations to the concerned agencies for future research and development.

5.1 Conclusions

Though three dimensional objects have existed for ages, it was only recently found necessary to study about it. Over the past decades many researches was carried out on this topic. CityGML is one kind of model resulted from this research, developed for exchange and storage of virtual 3D city objects. Due to its rich thematic classes and capability to extend through ADE mechanism, CityGML was adopted in diverse application fields. Accordingly, it was also adopted here to design an ADE for 3D cadastre.

UML model of CityGML ADE for 3D building cadastral database, which consist of two parts: UML modeling of CityGML ADE and its realization into relational database, are successfully implemented in this study. The classes such as ApartmentUnit, AccessoryUnit, CommonPropertyUnit, and LegalBuildingSpace needed to differentiate the legal spaces related to a building were defined and added as subclass of class CityObject of CityGML. Additionally, new attributes such as buildingNo and plotNo were added to class AbstractBuilding using the stereotyped

class <ADEElement>. These additions of new properties have added semantic information needed for 3D building cadastre in the classes of CityGML.

The ShapeChange software, which is an open-source Java tool for generating schemas and associated resources from a UML domain model, is here used to generate XML schema document and codelists dictionaries from the UML model of Cadastre ADE. ShapeChange can directly access Enterprise Architect models via its Java API and it provides a mechanism for providing arguments via a configuration file. This configuration file can be configured according to the user's requirements providing an easy way to generate the required result as per our convenience.

All the modeling phase from UML design to DDL generation for database is done in EA. Due to its comprehensive functionality for modeling; all the work can be done under one roof without the need to learn or process in other software. Moreover, EA has been chosen by ISO and OGC for the modeling standards in UML and the models are freely accessible in the internet. This has speed up the overall workflow.

Since the existing data were only available in 2D, 3D data construction is needed to be accomplished. For that building footprint and floor plan are used as an input data to manually construct 3D building model in Trimble SketchUp software. The resulted SketchUp 3D building model was translated to CityGML model using FME tool. The output from this translation is a GML file. Through these results, it was found that 3D building model with its detail can be efficiently constructed using SketchUp software; however, it is time consuming when there are many details to be modeled. Therefore, the detail of the building should be decided in advance before starting 3D building construction. Similarly, the GML translation of the detailed SketchUp model increases the number of transformers required making the FME

workflow more complex. Also in the beginning of the research, the translation from SketchUp model to CityGML and Cadastre ADE in FME was running into loop when executing. The problem was reported to support team of FME and it was found that the ADE generated from ShapeChange have mixture of GML 3.1.1 and 3.2.1 definition, which is not supported by FME. This problem was corrected by configuring the ShapeChange configuration file to generate XML schema for ADE in GML 3.1.1 definition.

Overall a prototype of 3D building cadastre was here successfully designed and developed using CityGML as a base module. The database implementation was done in the MS SQL Server 2008, which is the database system of NLCS. EA supports modeling of database schema and automatic generation of DDL scripts for other DBMS targets such as DB2, Firebird, MS Access (all versions), MySQL, Oracle (versions 9i, 10g, 11g and 12c), and PostgreSQL. Therefore, this implementation in MS SQL Server can be duplicated to other DBMS. Finally, it is concluded that this work can provide a basis for 3D cadastre implementation in Bhutan.

5.2 Recommendations

In this study, 3D building cadastral database design was examined and designed using various tools and models. The choice of tools and models has provided some merits and demerits to the result of this research. Considering the demerits and the opportunity sighted to improve this work, the following recommendations could be made for further studies:

(1) At present, many software packages for building 3D model are available in the market. They can provide various specific tools and visualization platform for

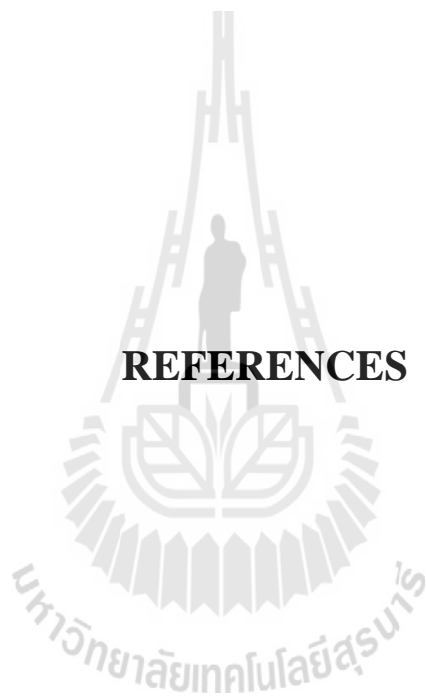
various types of construction and building with some limitation. Therefore, it requires reviewing specification of each software in detail before implementation of 3D cadastre in Bhutan.

(2) Basically any development of database requires a user interface for importing, exporting, editing, and querying the data. However, in this study due to time constraint, development of an interface could not be included. Therefore, a comprehensive interface should be design to work with the created prototype. Additionally, a study on the web application technology in this domain is recommended.

(3) Due to tedious and time consuming workflow of 3D building model preparation under SketchUp, NLCS shall discuss with the concerned agencies, particularly Thimphu Thromde, to investigate the possibility for submission of floor plan in 3D format while seeking building occupancy certificate approval. This can speed up acquisition of 3D building model to implement 3D cadastre in Bhutan. Other alternative is the automation of some of the workflow for 3d model construction in SketchUp, which is possible through the programming platform called Ruby console available in SketchUp. Recently there is a commercial SketchUp plug-in for CityGML named CityEditor was available in the market. Unfortunately, this plug-in came out late and cannot be explored or used in this research. Therefore, its functionality and applicability in this work shall be explored.

(4) In this work, only core and building modules of CityGML were used to design the 3D building cadastral database. Therefore, utilization of other modules of CityGML and other building model such as Building Information Models (BIM) as a basic layer in the implementation of 3D cadastre shall be more investigated.

REFERENCES



REFERENCES

- 3dcitydb. (2015). **3D City Database for CityGML Version 3.0.0**. [Online] Available: http://www.3dcitydb.net/3dcitydb/fileadmin/downloaddata/3DCityDB_Documentation_v3.pdf. Accessed date: November 21, 2015.
- Agilemodeling. (2015). [Online] Available: <http://www.agilemodeling.com/artifacts/classDiagram.html>. Accessed date: June 20, 2015.
- Aien, A., Kalantari, M., Rajabifard, A., Williamson, I., and Wallace, J. (2013). Towards integration of 3D legal and physical objects in cadastral data model. **Land Use Policy**. 35:140-154.
- Aien, A., Kalantari, M., Rajabifard, A., Williamson, I.P., and Shojaei, D. (2012). Developing and testing a 3D cadastral data model: A case study in Australia. **ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences**. I-4: 1-6.
- Andrade, A.J.B., Carneiro, A.F.T., and dos Santos, J.C. (2013). LADM Specification of a Relational Database for the Republic of Cape Verde. **5th Land Administration Domain Model Workshop**, Kuala Lumpur, Malaysia. 345-360.
- Budisusanto, Y., Aditya, T., and Muryamto, R. (2013). LADM implementation prototype for 3D cadastral information system of multi-level apartment in

- Indonesia. **International FIG workshop on the Land Administration Domain Model**, Kuala Lumpur, Malaysia. 1-11.
- Bydlosz, J. (2013). Towards LADM Country Cadastral Profile - Case Poland. **5th Land Administration Domain Model Workshop**, Kuala Lumpur, Malaysia. 247-260.
- Cagdas, V. (2013). An Application Domain Extension to CityGML for immovable property taxation: A Turkish case study. **International Journal of Applied Earth Observation and Geoinformation**. 21: 545-555.
- Chiang, H. (2012). Data modelling and application of 3D cadastral in Taiwan. **3rd International Workshop on 3D Cadastre**, Shenzhen, China. 137-157.
- Dsilva, M.G. (2009). **A feasibility study on CityGML for cadastral purposes**. Master Thesis. Eindhoven University of Technology, the Netherlands.
- Elia, E.A., Zevenbergen, J.A., Lemmen, C.H.J., and van Oosterom, P.J.M. (2011). The land administration domain model as the reference model for the Cyprus land information system. **Survey Review**. 45(329): 100-110.
- Gózdź, K., Pachelski, W., van Oosterom, P., and Coors, V. (2014). The Possibilities of Using CityGML for 3D Representation of Buildings in the Cadastre. **4th International Workshop on 3D Cadastres**, Dubai, United Arab Emirates. 339-361.
- Hendriatiningsih, S., Abdulharis, R., and Hernandi, A. (2012). Revisiting the Concept of Boundary on 3D Cadastre in Indonesia. **FIG Working Week 2012**, Rome, Italy. 1-9.

Hespanha, J.P. (2012). **Development Methodology for an Integrated Legal Cadastre: Deriving Portugal Country Model from the Land Administration Domain Model**. PhD Thesis. Delft University of Technology, the Netherlands.

ISO/TC 211. (2012). Geographic information - Land Administration Domain Model: ISO/FDIS 19152. 1-118.

Kalantari, M. (2008). **Cadastral Data Modelling – A Tool for e-Land Administration**. PhD Thesis. The University of Melbourne, Australia.

Karpina. (2014). A 3D CityGML based GIS system for shallow water simulations.

Kim, Y., Kang, H., and Lee, J. (2013). Development of Indoor Spatial Data Model Using CityGML ADE. **ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences**. XL-2/W2: 41-45.

kuenselonline. (2015). [Online] Available: <http://www.kuenselonline.com>. Accessed date: May 29, 2015.

Lemmen, C.H.J., and van Oosterom, P.J.M. (2013). The Land Administration Domain Model Standard. **5th Land Administration Domain Model Workshop**, Kuala Lumpur. 11-30.

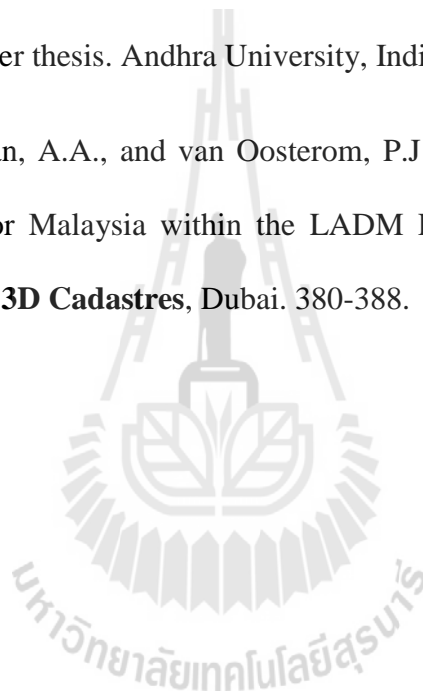
Lemmen, C.H.J., van Oosterom, P.J.M., Uitermark, H.T., Zevenbergen, J.A., and Cooper, A.K. (2011). Interoperable domain models: The ISO Land Administration Domain Model LADM and its external classes. **28th Urban Data Management Symposium**, Delft, the Netherlands. 31-39.

- Mingru, J. (2007). **Registration of 3D cadastral Objects in China**. Master Thesis. International Institute for Geo-Information Science and Earth Observation, the Netherlands.
- NLC. (2014). Sazhi. **National Land Commission's quarterly newsletter**. 1(1): 1-14.
- OGC. (2012). OGC City Geography Marking Language (CityGML) Encoding Standard.
- OGC. (2014). Modeling an application domain extension of CityGML in UML OGC Best Practice.
- El-Mrkawy, M., and et Östman, A. 2012. Feasibility of Building Information Models for 3D Cadastre in Unified City Models. **International Journal of E-Planning Research**. 1/4: 35-58
- Panchal, H., Khan, R., Sengupta, S., and Sarda, N.L. (2011). GIS-based Smart Campus System using 3D Modeling. **Geospatial World Forum**, Hyderabad, India.
- Park, S., Lee, J., and Li, H. (2009). 3D Cadastre Data Model in Korea; based on case studies in Seoul. **The Journal of GIS Association of Korea**. 17/4:469-481.
- Ronsdorf, C., Wilson, D., and Stoter, J. (2014). Integration of Land Administration Domain Model with CityGML for 3D Cadastre. **4th International Workshop on 3D Cadastres**, Dubai. 313-322.
- shapechange. (2015). [Online] Available: www.shapechange.net. Accessed date: June 03, 2015.

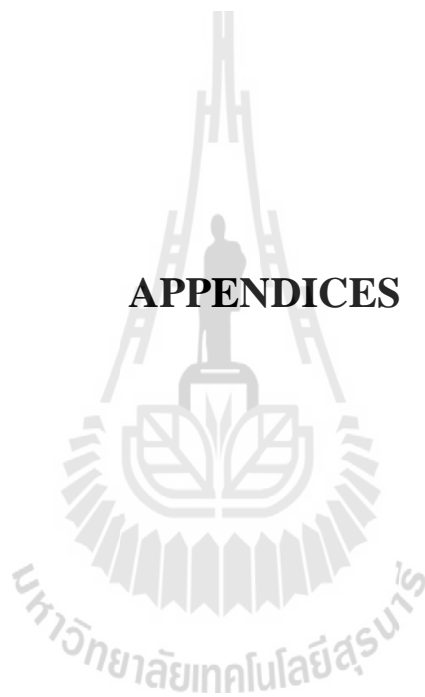
- Shen, Y., Lin, L., and Renzhong, G. (2011). Building 3D cadastral system based on 2D survey plans with SketchUp. **Geo-spatial Information Science**. 14(2):129-136.
- Snowflake Software. (2015). **Developing CityGML ADEs**. [Online] Available: <https://wiki.snowflakesoftware.com/download/attachments/5505036/DevelopingCityGMLADE.pptx?api=v2>. Accessed date: March 15, 2015.
- Sparks, G. (2015). **Database Modelling in UML**. [Online] Available: <http://www.sparkxsystems.com.au>. Accessed date: May 30, 2015.
- Sparkx Systems. (2011). **Data Modeling: From conceptual model to DBMS**. [Online] Available: <http://www.sparkxsystems.com>. Accessed date: April 21, 2015.
- Stadler, A., and Kolbe, T.H. (2007). Spatio-semantic coherence in the integration of 3D city models. **5th International ISPRS Symposium on Spatial Data Quality**, Enschede, the Netherlands.
- Stoter, J., Beetz, J., Ledoux, H., Reuvers, M., Klooster, R., Janssen, P., Penninga, F., Zlatanova, S., and van den Brink, L. (2013). Implementation of a national 3D standard: Case of The Netherlands. **Progress and New Trends in 3D Geoinformation Sciences**. Berlin, Springer-Verlag Lecture Notes in Geoinformation and Cartography. 277–298.
- Stoter, J.E. (2004). **3D Cadastre**. PhD Thesis. Delft University of Technology, the Netherlands.

- Stoter, J.E. and van Oosterom, P. (2006). 3D Cadastre in an International Context: Legal, Organizational, and Technological Aspects. **CRC Press**. 1-344.
- Sucaya, I.K.G.A. (2009). **Application and validation of land administration domain model in a real life situation (A case study in Indonesia)**. Master Thesis. International Institute for Geo-Information Science and Earth Observation, the Netherlands.
- Swedesurvey AB. (2003). **Strengthening National Geo-Information Management System**. NLCS, Bhutan.
- thimphucity. (2015). [Online] Available: <http://www.thimphucity.bt>. Accessed date: June 02, 2015.
- van den Brink, L., Stoter, J., and Zlatanova, S. (2012). Modelling an Application Domain Extension of CityGML in UML. **International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences**. XXXVIII-4/C26: 11-14.
- van den Brink, L., Stoter, J., and Zlatanova, S. (2013). UML-based approach to developing a CityGML application domain extension. **Transactions in GIS**. 17(6): 920-942.
- van Oosterom, P. (2014). Survey of Israel Three-Dimensional Cadastre and the ISO 19152 - The Land Administration Domain Model. **Delft University of Technology**, the Netherlands.

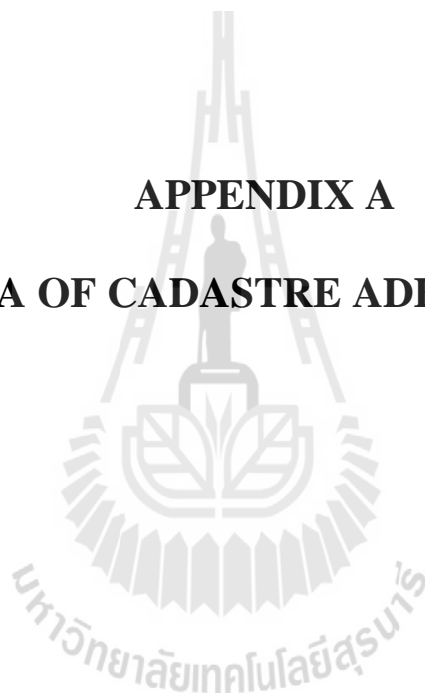
- van Oosterom, P., Stoter, J., and Lemmen, C. (2005). **Modelling of 3D Cadastral Systems**. [Online] Available: http://www.gdmc.nl/publications/2005/3D_cadastral_systems.pdf. Accessed date: February 02, 2016.
- Wang, L., and Sohn, G. (2011). An integrated framework for reconstructing full 3D building models. **Advances in 3D Geo-Information Sciences**. 261-274.
- Wate, P.S. (2014). **3D GIS modeling at semantic level using CityGML for urban segment**. Master thesis. Andhra University, India.
- Zulkifli, N.A., Rahman, A.A., and van Oosterom, P.J.M. (2014). 3D Strata Objects Registration for Malaysia within the LADM Framework. **4th International Workshop on 3D Cadastres**, Dubai. 380-388.



APPENDICES



APPENDIX A
XML SCHEMA OF CADASTRE ADE AND CODELISTS



```

<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:bldg="http://www.opengis.net/citygml/building/2.0" xmlns:cad="file:///C:/ShapeChange/output/CadastreADESchema"
xmlns:core="http://www.opengis.net/citygml/2.0" xmlns:gml="http://www.opengis.net/gml" targetNamespace="file:///C:/ShapeChange/output/CadastreADESchema"
elementFormDefault="qualified" version="1.0">
  <import namespace="http://www.opengis.net/citygml/2.0" schemaLocation="http://schemas.opengis.net/citygml/2.0/cityGMLBase.xsd"/>
  <import namespace="http://www.opengis.net/citygml/building/2.0" schemaLocation="http://schemas.opengis.net/citygml/building/2.0/building.xsd"/>
  <import namespace="http://www.opengis.net/gml" schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
  <!--XML Schema document created by ShapeChange - http://shapechange.net/-->
  <element name="buildingNo" type="string" substitutionGroup="bldg:_GenericApplicationPropertyOfAbstractBuilding"/>
  <element name="plotNo" type="string" substitutionGroup="bldg:_GenericApplicationPropertyOfAbstractBuilding"/>
  <element name="legalBu" type="cad:LegalSpaceBuildingPropertyType" substitutionGroup="bldg:_GenericApplicationPropertyOfAbstractBuilding"/>
  <element name="AccessoryUnit" type="cad:AccessoryUnitType" substitutionGroup="cad:LegalSpaceBuildingUnit"/>
  <complexType name="AccessoryUnitType">
    <complexContent>
      <extension base="cad:LegalSpaceBuildingUnitType">
        <sequence>
          <element name="accType" type="gml:CodeType"/>
          <element name="apUnit2" type="cad:apartmentUnitPropertyType" maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="AccessoryUnitPropertyType">
    <sequence minOccurs="0">
      <element ref="cad:AccessoryUnit"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
  </complexType>
  <element name="buildingInstID" type="string" substitutionGroup="bldg:_GenericApplicationPropertyOfBuildingInstallation"/>
  <element name="cpUnit2" type="cad:CommonPropertyUnitPropertyType" substitutionGroup="bldg:_GenericApplicationPropertyOfBuildingInstallation"/>
  <element name="CommonPropertyUnit" type="cad:CommonPropertyUnitType" substitutionGroup="cad:LegalSpaceBuildingUnit"/>
  <complexType name="CommonPropertyUnitType">
    <complexContent>
      <extension base="cad:LegalSpaceBuildingUnitType">
        <sequence>
          <element name="cpuType" type="gml:CodeType"/>
          <element name="bi1" type="bldg:BuildingInstallationPropertyType" maxOccurs="unbounded"/>
          <element name="apUnit1" type="cad:apartmentUnitPropertyType" maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>

```

Figure A-1 Cadastre ADE.xsd.

```

<complexType name="CommonPropertyUnitPropertyType">
  <sequence minOccurs="0">
    <element ref="cad:CommonPropertyUnit"/>
  </sequence>
  <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
<element name="LegalSpaceBuilding" type="cad:LegalSpaceBuildingType" substitutionGroup="cad:LegalSpaceBuildingUnit"/>
<complexType name="LegalSpaceBuildingType">
  <complexContent>
    <extension base="cad:LegalSpaceBuildingUnitType">
      <sequence>
        <element name="lsbID" type="string"/>
        <element name="absBu" type="bldg:AbstractBuildingType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="LegalSpaceBuildingPropertyType">
  <sequence minOccurs="0">
    <element ref="cad:LegalSpaceBuilding"/>
  </sequence>
  <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
<element name="LegalSpaceBuildingUnit" type="cad:LegalSpaceBuildingUnitType" abstract="true" substitutionGroup="cad:SpatialUnit"/>
<complexType name="LegalSpaceBuildingUnitType" abstract="true">
  <complexContent>
    <extension base="cad:SpatialUnitType">
      <sequence>
        <element name="room1" type="bldg:RoomType" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="LegalSpaceBuildingUnitPropertyType">
  <sequence minOccurs="0">
    <element ref="cad:LegalSpaceBuildingUnit"/>
  </sequence>
  <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>

```

Figure A-1 (Continued).

```

<element name="roomID" type="string" substitutionGroup="bldg:GenericApplicationPropertyOfRoom"/>
<element name="lsbUnit1" type="cad:LegalSpaceBuildingUnitPropertyType" substitutionGroup="bldg:GenericApplicationPropertyOfRoom"/>
<element name="SpatialUnit" type="cad:SpatialUnitType" abstract="true" substitutionGroup="core:_CityObject"/>
<complexType name="SpatialUnitType" abstract="true">
  <complexContent>
    <extension base="core:AbstractCityObjectType">
      <sequence>
        <element name="suID" type="string"/>
        <element name="area" type="gml:AreaType"/>
        <element name="volume" type="gml:VolumeType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="SpatialUnitPropertyType">
  <sequence minOccurs="0">
    <element ref="cad:SpatialUnit"/>
  </sequence>
  <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
<element name="apartmentUnit" type="cad:apartmentUnitType" substitutionGroup="cad:LegalSpaceBuildingUnit"/>
<complexType name="apartmentUnitType">
  <complexContent>
    <extension base="cad:LegalSpaceBuildingUnitType">
      <sequence>
        <element name="unitNo" type="string"/>
        <element name="roomCount" type="integer"/>
        <element name="floorNo" type="gml:CodeType"/>
        <element name="accUnit1" type="cad:AccessoryUnitPropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <element name="cpUnit1" type="cad:CommonPropertyUnitPropertyType" minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="apartmentUnitPropertyType">
  <sequence minOccurs="0">
    <element ref="cad:apartmentUnit"/>
  </sequence>
  <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
</schema>

```

Figure A-1 (Continued).

```

AccessoryUnitType.xml
1 <?xml-stylesheet type='text/xsl' href='../CodelistDictionary-v32.xsl'?><Dictionary xmlns="http://www.opengis.net/gml"
2 xmlns:gml="http://www.opengis.net/gml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3 gml:id="AccessoryUnitType" xsi:schemaLocation="http://www.opengis.net/gml null">
4 <description>
5 </description>
6 <identifier codeSpace="file:///C:/ShapeChange/output/CadastreADESchema">AccessoryUnitType</identifier>
7 <dictionaryEntry>
8 <Definition gml:id="_7651_9624">
9 <description>
10 </description>
11 <identifier codeSpace="">Garage</identifier>
12 </Definition>
13 </dictionaryEntry>
14 <dictionaryEntry>
15 <Definition gml:id="_7651_9625">
16 <description>
17 </description>
18 <identifier codeSpace="">Storeroom</identifier>
19 </Definition>
20 </dictionaryEntry>
21 </Dictionary>

```

Figure A-2 Codelist of accessory unit in xml.

```

InstallationType.xml BuildingType.xml CommonPropertyUnitType.xml FloorNo.xml LegalPropertyType.xml RoofType.xml
<?xml-stylesheet type='text/xsl' href='../CodelistDictionary-v32.xsl'?><Dictionary
xmlns="http://www.opengis.net/gml" xmlns:gml="http://www.opengis.net/gml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" gml:id="LegalPropertyType"
xsi:schemaLocation="http://www.opengis.net/gml null">
<description>
</description>
<identifier codeSpace="file:///C:/ShapeChange/output/CadastreADESchema">LegalPropertyType</identifier>
<dictionaryEntry>
<Definition gml:id="_22983_28972">
<description>
</description>
<identifier codeSpace="">ApartmentUnit</identifier>
</Definition>
</dictionaryEntry>
<dictionaryEntry>
<Definition gml:id="_22983_28973">
<description>
</description>
<identifier codeSpace="">CommonPropertyUnit</identifier>
</Definition>
</dictionaryEntry>
<dictionaryEntry>
<Definition gml:id="_22983_28974">
<description>
</description>
<identifier codeSpace="">AccessoryUnit</identifier>
</Definition>
</dictionaryEntry>
</Dictionary>

```

Figure A-3 Codelist of legal property type in xml.

```

<?xml-stylesheet type='text/xsl' href='./CodelistDictionary-v32.xsl'?><Dictionary xmlns="http://www.opengis.net/gml"
xmlns:gml="http://www.opengis.net/gml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" gml:id="FloorNo"
xsi:schemaLocation="http://www.opengis.net/gml null">
  <description>
  </description>
  <identifier codeSpace="file:///C:/ShapeChange/output/CadastreADESchema">FloorNo</identifier>
  <dictionaryEntry>
    <Definition gml:id="_7652_9626">
      <description>
      </description>
      <identifier codeSpace="">1</identifier>
    </Definition>
  </dictionaryEntry>
  <dictionaryEntry>
    <Definition gml:id="_7652_9627">
      <description>
      </description>
      <identifier codeSpace="">2</identifier>
    </Definition>
  </dictionaryEntry>
  <dictionaryEntry>
    <Definition gml:id="_7652_9628">
      <description>
      </description>
      <identifier codeSpace="">3</identifier>
    </Definition>
  </dictionaryEntry>
  <dictionaryEntry>
    <Definition gml:id="_7652_9629">
      <description>4th Floor</description>
      <identifier codeSpace="">4</identifier>
    </Definition>
  </dictionaryEntry>
  <dictionaryEntry>
    <Definition gml:id="_7652_9630">
      <description>5th Floor</description>
      <identifier codeSpace="">5</identifier>
    </Definition>
  </dictionaryEntry>
  <dictionaryEntry>
    <Definition gml:id="_7652_9631">
      <description>6th Floor</description>
      <identifier codeSpace="">6</identifier>
    </Definition>
  </dictionaryEntry>
  <dictionaryEntry>
    <Definition gml:id="_7652_28531">
      <description>UG = Underground</description>
      <identifier codeSpace="">UG</identifier>
    </Definition>
  </dictionaryEntry>
</Dictionary>

```

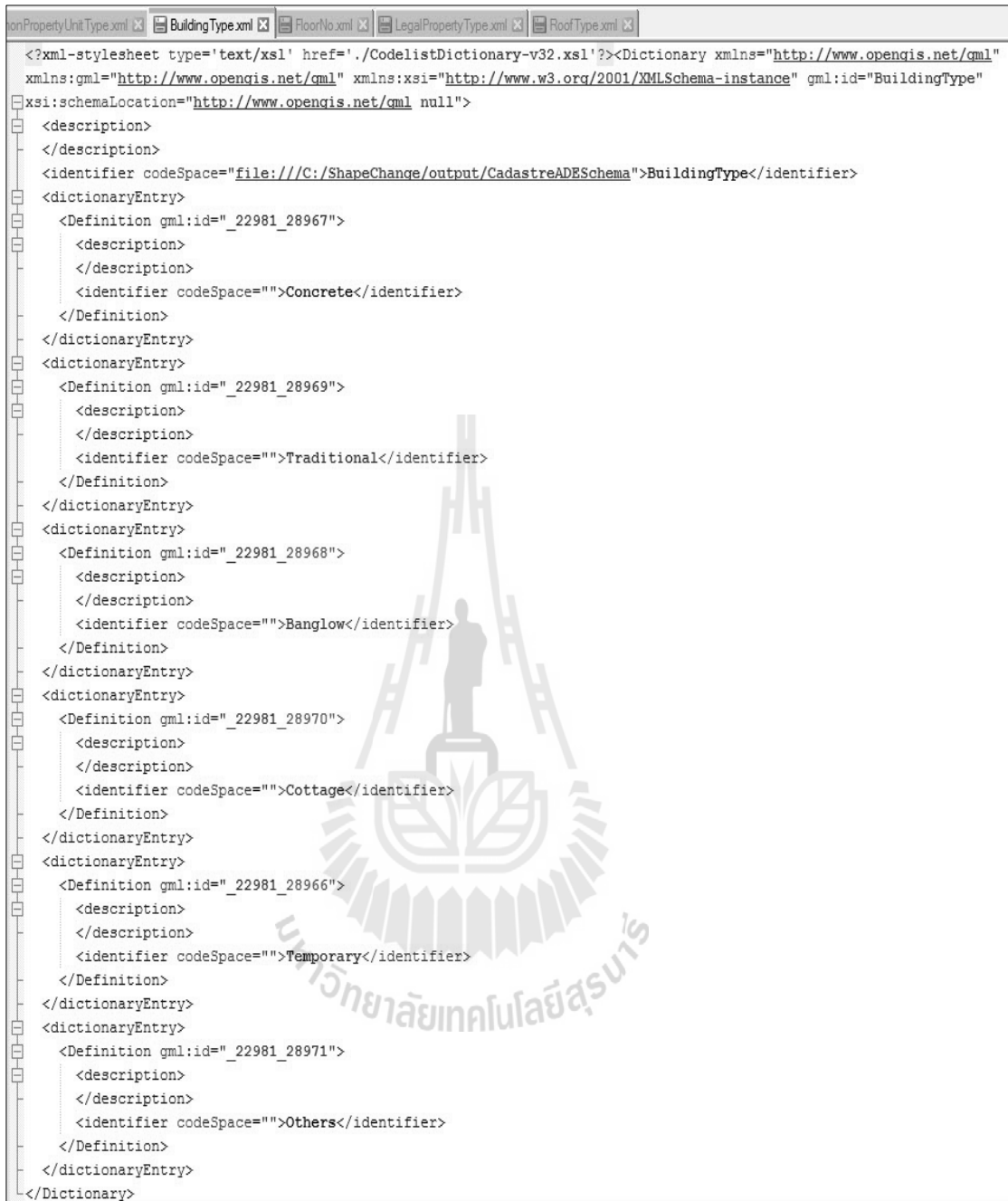
Figure A-4 Codelist of floor number in xml.

```

<?xml-stylesheet type='text/xsl' href='../CodelistDictionary-v32.xsl'?><Dictionary xmlns="http://www.opengis.net/gml"
xmlns:gml="http://www.opengis.net/gml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" gml:id="RoofType"
xsi:schemaLocation="http://www.opengis.net/gml null">
  <description>
  </description>
  <identifier codeSpace="file:///C:/ShapeChange/output/CadastreADESchema">RoofType</identifier>
  <dictionaryEntry>
    <Definition gml:id="_22979_28960">
      <description>
      </description>
      <identifier codeSpace="">Concrete</identifier>
    </Definition>
  </dictionaryEntry>
  <dictionaryEntry>
    <Definition gml:id="_22979_28961">
      <description>
      </description>
      <identifier codeSpace="">Shingle</identifier>
    </Definition>
  </dictionaryEntry>
  <dictionaryEntry>
    <Definition gml:id="_22979_28962">
      <description>
      </description>
      <identifier codeSpace="">Metal roofing</identifier>
    </Definition>
  </dictionaryEntry>
  <dictionaryEntry>
    <Definition gml:id="_22979_28963">
      <description>
      </description>
      <identifier codeSpace="">Slate</identifier>
    </Definition>
  </dictionaryEntry>
</Dictionary>

```

Figure A-5 Codelist of roof type in xml.



```

<?xml-stylesheet type='text/xsl' href='./CodelistDictionary-v32.xsl'?><Dictionary xmlns="http://www.opengis.net/gml"
xmlns:gml="http://www.opengis.net/gml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" gml:id="BuildingType"
xsi:schemaLocation="http://www.opengis.net/gml null">
  <description>
  </description>
  <identifier codeSpace="file:///C:/ShapeChange/output/CadastreADESchema">BuildingType</identifier>
  <dictionaryEntry>
    <Definition gml:id="_22981_28967">
      <description>
      </description>
      <identifier codeSpace="">Concrete</identifier>
    </Definition>
  </dictionaryEntry>
  <dictionaryEntry>
    <Definition gml:id="_22981_28969">
      <description>
      </description>
      <identifier codeSpace="">Traditional</identifier>
    </Definition>
  </dictionaryEntry>
  <dictionaryEntry>
    <Definition gml:id="_22981_28968">
      <description>
      </description>
      <identifier codeSpace="">Banglow</identifier>
    </Definition>
  </dictionaryEntry>
  <dictionaryEntry>
    <Definition gml:id="_22981_28970">
      <description>
      </description>
      <identifier codeSpace="">Cottage</identifier>
    </Definition>
  </dictionaryEntry>
  <dictionaryEntry>
    <Definition gml:id="_22981_28966">
      <description>
      </description>
      <identifier codeSpace="">Temporary</identifier>
    </Definition>
  </dictionaryEntry>
  <dictionaryEntry>
    <Definition gml:id="_22981_28971">
      <description>
      </description>
      <identifier codeSpace="">Others</identifier>
    </Definition>
  </dictionaryEntry>
</Dictionary>

```

Figure A-6 Codelist of building type in xml.

APPENDIX B
THE SQL SERVER 2008 DATA TYPES

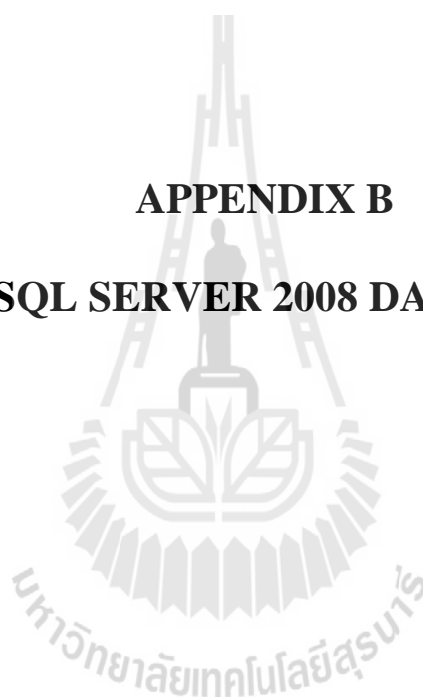


Table B-1 String types.

Data type	Description	Storage
char(n)	Fixed width character string. Maximum 8,000 characters	Defined width
varchar(n)	Variable width character string. Maximum 8,000 characters	2 bytes + number of chars
varchar(max)	Variable width character string. Maximum 1,073,741,824 characters	2 bytes + number of chars
text	Variable width character string. Maximum 2GB of text data	4 bytes + number of chars
nchar	Fixed width Unicode string. Maximum 4,000 characters	Defined width x 2
nvarchar	Variable width Unicode string. Maximum 4,000 characters	
nvarchar(max)	Variable width Unicode string. Maximum 536,870,912 characters	
ntext	Variable width Unicode string. Maximum 2GB of text data	
bit	Allows 0, 1, or NULL	
binary(n)	Fixed width binary string. Maximum 8,000 bytes	
varbinary	Variable width binary string. Maximum 8,000 bytes	
varbinary(max)	Variable width binary string. Maximum 2GB	
image	Variable width binary string. Maximum 2GB	

Table B-2 Number types.

Data type	Description	Storage
tinyint	Allows whole numbers from 0 to 255	1 byte
smallint	Allows whole numbers between -32,768 and 32,767	2 bytes
int	Allows whole numbers between -2,147,483,648 and 2,147,483,647	4 bytes
bigint	Allows whole numbers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807	8 bytes
	Fixed precision and scale numbers.	
	Allows numbers from $-10^{38} + 1$ to $10^{38} - 1$.	
decimal(p,s)	The p parameter indicates the maximum total number of digits that can be stored (both to the left and to the right of the decimal point). p must be a value from 1 to 38. Default is 18.	5-17 bytes
	The s parameter indicates the maximum number of digits stored to the right of the decimal point. s must be a value from 0 to p. Default value is 0	

Table B-2 (Continued).

Data type	Description	Storage
	Fixed precision and scale numbers.	
	Allows numbers from $-10^{38} + 1$ to $10^{38} - 1$.	
numeric(p,s)	The p parameter indicates the maximum total number of digits that can be stored (both to the left and to the right of the decimal point). p must be a value from 1 to 38. Default is 18.	5-17 bytes
	The s parameter indicates the maximum number of digits stored to the right of the decimal point. s must be a value from 0 to p. Default value is 0	
smallmoney	Monetary data from -214,748.3648 to 214,748.3647	4 bytes
money	Monetary data from -922,337,203,685,477.5808 to 922,337,203,685,477.5807	8 bytes
	Floating precision number data from $-1.79E + 308$ to $1.79E + 308$.	
float(n)	The n parameter indicates whether the field should hold 4 or 8 bytes. float(24) holds a 4-byte field and float(53) holds an 8-byte field. Default value of n is 53.	4 or 8 bytes
real	Floating precision number data from $-3.40E + 38$ to $3.40E + 38$	4 bytes

Table B-3 Date types.

Data type	Description	Storage
datetime	From January 1, 1753 to December 31, 9999 with an accuracy of 3.33 milliseconds	8 bytes
datetime2	From January 1, 0001 to December 31, 9999 with an accuracy of 100 nanoseconds	6-8 bytes
smalldatetime	From January 1, 1900 to June 6, 2079 with an accuracy of 1 minute	4 bytes
date	Store a date only. From January 1, 0001 to December 31, 9999	3 bytes
time	Store a time only to an accuracy of 100 nanoseconds	3-5 bytes
datetimeoffset	The same as datetime2 with the addition of a time zone offset	8-10 bytes
timestamp	Stores a unique number that gets updated every time a row gets created or modified. The timestamp value is based upon an internal clock and does not correspond to real time. Each table may have only one timestamp variable	

Table B-4 Other data types.

Data type	Description
sql_variant	Stores up to 8,000 bytes of data of various data types, except text, ntext, and timestamp
uniqueidentifier	Stores a globally unique identifier (GUID)
xml	Stores XML formatted data. Maximum 2GB
cursor	Stores a reference to a cursor used for database operations
table	Stores a result-set for later processing

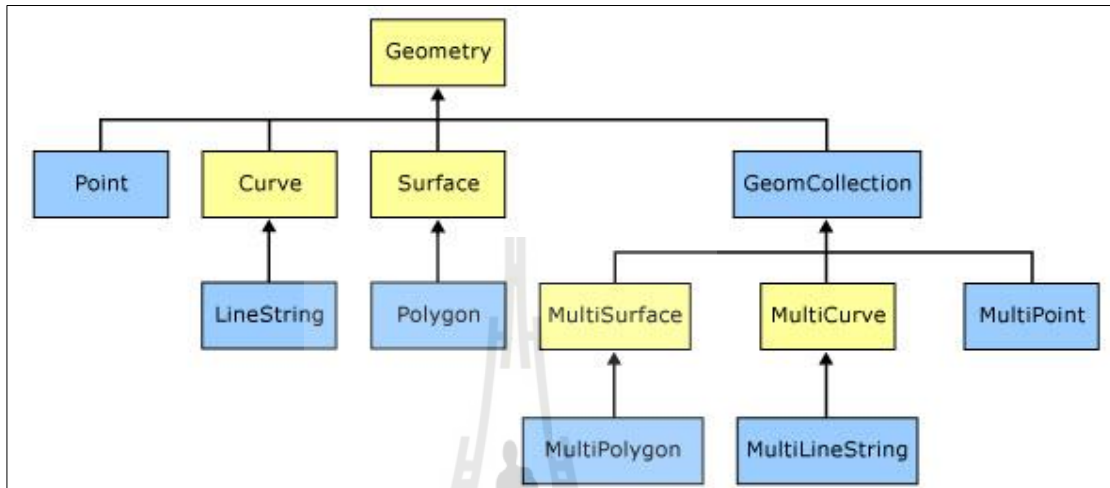
B-5 Spatial Data Types

There are two types of spatial data such as:

- (1) The **geometry** data type supports planar, or Euclidean (flat-earth), data.
- (2) The **geometry** data type both conforms to the Open Geospatial Consortium (OGC) Simple Features for SQL Specification version 1.1.0 and is compliant with SQL MM (ISO standard).

The **geometry** and **geography** data types support sixteen spatial data objects, or instance types. However, only eleven of these instance types are *instantiable*; you can create and work with these instances (or instantiate them) in a database. The ten instantiable types of the geometry and geography data types are Point, MultiPoint, LineString, CircularString, MultiLineString, CompoundCurve, Polygon, CurvePolygon, MultiPolygon, and GeometryCollection. There is one additional instantiable type for the geography data type: FullGlobe.

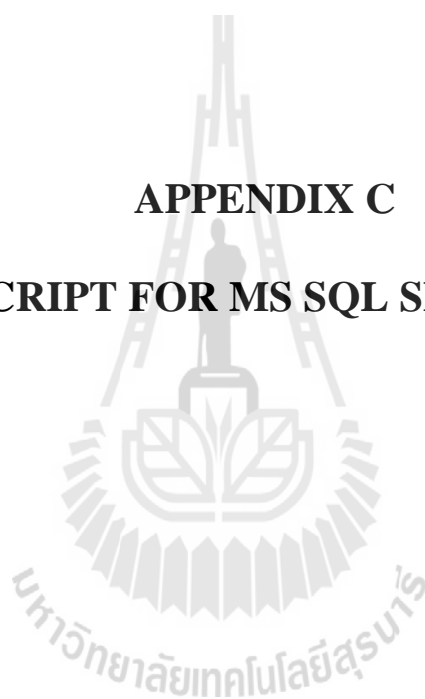
The figure below depicts the geometry hierarchy upon which the geometry and geography data types are based. The instantiable types of geometry and geography are indicated in blue.



Simple types	
Point	zero-dimensional object representing a single location and may contain Z (elevation) and M (measure) values
LineString	one-dimensional object representing a sequence of points and the line segments connecting them
CircularString	collection of zero or more continuous circular arc segments (a curved segment defined by three points in a two-dimensional plane)
CompoundCurve	collection of zero or more continuous CircularString or LineString
Polygon	two-dimensional surface stored as a sequence of points defining an exterior bounding ring and zero or more interior rings
CurvePolygon	topologically closed surface defined by an exterior bounding ring and zero or more interior rings
Collective types	
MultiPoint	collection of zero or more points
MultiLineString	collection of zero or more geometry or geography LineString instances
MultiPolygon	collection of zero or more Polygon instances
GeometryCollection	collection of zero or more geometry or geography instances

APPENDIX C

DDL SCRIPT FOR MS SQL SERVER 2008




```

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_AccessoryUnit_ApartmentUnit]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [AccessoryUnit] DROP CONSTRAINT [FK_AccessoryUnit_ApartmentUnit]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_AccessoryUnit_Building]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [AccessoryUnit] DROP CONSTRAINT [FK_AccessoryUnit_Building]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_ApartmentUnit_Room]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [ApartmentUnit] DROP CONSTRAINT [FK_ApartmentUnit_Room]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_Building_Building]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [Building] DROP CONSTRAINT [FK_Building_Building]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_Building_Building_02]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [Building] DROP CONSTRAINT [FK_Building_Building_02]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_Building_CityObject]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [Building] DROP CONSTRAINT [FK_Building_CityObject]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_Building_SurfaceGeometry]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [Building] DROP CONSTRAINT [FK_Building_SurfaceGeometry]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_Building_SurfaceGeometry_02]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [Building] DROP CONSTRAINT [FK_Building_SurfaceGeometry_02]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_Building_SurfaceGeometry_03]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [Building] DROP CONSTRAINT [FK_Building_SurfaceGeometry_03]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_Building_SurfaceGeometry_04]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [Building] DROP CONSTRAINT [FK_Building_SurfaceGeometry_04]
GO

```

Figure C-1 Drop foreign key (FK) constraints.

```
IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_Building_SurfaceGeometry_05]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [Building] DROP CONSTRAINT [FK_Building_SurfaceGeometry_05]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_BuildingInstallation_Building]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [BuildingInstallation] DROP CONSTRAINT [FK_BuildingInstallation_Building]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_BuildingInstallation_CityObject]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [BuildingInstallation] DROP CONSTRAINT [FK_BuildingInstallation_CityObject]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_BuildingInstallation_ImplicitGeometry]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [BuildingInstallation] DROP CONSTRAINT [FK_BuildingInstallation_ImplicitGeometry]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_BuildingInstallation_ObjectClass]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [BuildingInstallation] DROP CONSTRAINT [FK_BuildingInstallation_ObjectClass]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_BuildingInstallation_SurfaceGeometry]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [BuildingInstallation] DROP CONSTRAINT [FK_BuildingInstallation_SurfaceGeometry]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_BuildingInstallation_Room]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [BuildingInstallation] DROP CONSTRAINT [FK_BuildingInstallation_Room]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_CityObject_ObjectClass]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [CityObject] DROP CONSTRAINT [FK_CityObject_ObjectClass]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_CityObjectGroup_CityObject_02]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [CityObjectGroup] DROP CONSTRAINT [FK_CityObjectGroup_CityObject_02]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_CityObjectGroup_SurfaceGeometry]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [CityObjectGroup] DROP CONSTRAINT [FK_CityObjectGroup_SurfaceGeometry]
GO
```

Figure C-1 (Continued).

```
IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_CityObjectGroup_CityObject]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [CityObjectGroup] DROP CONSTRAINT [FK_CityObjectGroup_CityObject]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_CityObjectMember_CityModel]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [CityObjectMember] DROP CONSTRAINT [FK_CityObjectMember_CityModel]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_CityObjectMember_CityObject]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [CityObjectMember] DROP CONSTRAINT [FK_CityObjectMember_CityObject]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_OtherPropertyUnit_ApartmentUnit]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [CommonPropertyUnit] DROP CONSTRAINT [FK_OtherPropertyUnit_ApartmentUnit]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_OtherPropertyUnit_BuildingInstallation]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [CommonPropertyUnit] DROP CONSTRAINT [FK_OtherPropertyUnit_BuildingInstallation]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_OtherPropertyUnit_Room]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [CommonPropertyUnit] DROP CONSTRAINT [FK_OtherPropertyUnit_Room]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_GroupToCityObject_CityObject]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [GroupToCityObject] DROP CONSTRAINT [FK_GroupToCityObject_CityObject]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_GroupToCityObject_CityObjectGroup]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [GroupToCityObject] DROP CONSTRAINT [FK_GroupToCityObject_CityObjectGroup]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_ImplicitGeometry_SurfaceGeometry]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [ImplicitGeometry] DROP CONSTRAINT [FK_ImplicitGeometry_SurfaceGeometry]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_ObjectClass_ObjectClass]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [ObjectClass] DROP CONSTRAINT [FK_ObjectClass_ObjectClass]
GO
```

Figure C-1 (Continued).

```
IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_Opening_CityObject]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [Opening] DROP CONSTRAINT [FK_Opening_CityObject]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_Opening_ImplicitGeometry]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [Opening] DROP CONSTRAINT [FK_Opening_ImplicitGeometry]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_Opening_ObjectClass]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [Opening] DROP CONSTRAINT [FK_Opening_ObjectClass]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_Opening_SurfaceGeometry]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [Opening] DROP CONSTRAINT [FK_Opening_SurfaceGeometry]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_OpeningToBoundarySurface_BoundarySurface]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [OpeningToBoundarySurface] DROP CONSTRAINT [FK_OpeningToBoundarySurface_BoundarySurface]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_OpeningToBoundarySurface_Opening]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [OpeningToBoundarySurface] DROP CONSTRAINT [FK_OpeningToBoundarySurface_Opening]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_Room_SurfaceGeometry]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [Room] DROP CONSTRAINT [FK_Room_SurfaceGeometry]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_Room_SurfaceGeometry_02]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [Room] DROP CONSTRAINT [FK_Room_SurfaceGeometry_02]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_Room_Building]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [Room] DROP CONSTRAINT [FK_Room_Building]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FK_Room_CityObject]') AND OBJECTPROPERTY(id, 'IsForeignKey') = 1)
ALTER TABLE [Room] DROP CONSTRAINT [FK_Room_CityObject]
GO
```

Figure C-1 (Continued).

```
IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[AccessoryUnit]') AND OBJECTPROPERTY(id, 'IsUserTable') = 1)
DROP TABLE [AccessoryUnit]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[AccessoryUnitType]') AND OBJECTPROPERTY(id, 'IsUserTable') = 1)
DROP TABLE [AccessoryUnitType]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[ApartmentUnit]') AND OBJECTPROPERTY(id, 'IsUserTable') = 1)
DROP TABLE [ApartmentUnit]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[Building]') AND OBJECTPROPERTY(id, 'IsUserTable') = 1)
DROP TABLE [Building]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[BuildingInstallation]') AND OBJECTPROPERTY(id, 'IsUserTable') = 1)
DROP TABLE [BuildingInstallation]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[BuildingType]') AND OBJECTPROPERTY(id, 'IsUserTable') = 1)
DROP TABLE [BuildingType]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[CityModel]') AND OBJECTPROPERTY(id, 'IsUserTable') = 1)
DROP TABLE [CityModel]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[CityModelType]') AND OBJECTPROPERTY(id, 'IsUserTable') = 1)
DROP TABLE [CityModelType]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[CityObject]') AND OBJECTPROPERTY(id, 'IsUserTable') = 1)
DROP TABLE [CityObject]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[CityObjectGroup]') AND OBJECTPROPERTY(id, 'IsUserTable') = 1)
DROP TABLE [CityObjectGroup]
GO
```

Figure C-2 Drop tables.

```
IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[CityObjectMember]') AND OBJECTPROPERTY(id, 'IsUserTable') = 1)
DROP TABLE [CityObjectMember]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[CommonPropertyUnit]') AND OBJECTPROPERTY(id, 'IsUserTable') = 1)
DROP TABLE [CommonPropertyUnit]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[CommonPropertyUnitType]') AND OBJECTPROPERTY(id, 'IsUserTable') = 1)
DROP TABLE [CommonPropertyUnitType]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[FloorNo]') AND OBJECTPROPERTY(id, 'IsUserTable') = 1)
DROP TABLE [FloorNo]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[GroupToCityObject]') AND OBJECTPROPERTY(id, 'IsUserTable') = 1)
DROP TABLE [GroupToCityObject]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[ImplicitGeometry]') AND OBJECTPROPERTY(id, 'IsUserTable') = 1)
DROP TABLE [ImplicitGeometry]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[LegalPropertyType]') AND OBJECTPROPERTY(id, 'IsUserTable') = 1)
DROP TABLE [LegalPropertyType]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[ObjectClass]') AND OBJECTPROPERTY(id, 'IsUserTable') = 1)
DROP TABLE [ObjectClass]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[Opening]') AND OBJECTPROPERTY(id, 'IsUserTable') = 1)
DROP TABLE [Opening]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[OpeningToBoundarySurface]') AND OBJECTPROPERTY(id, 'IsUserTable') = 1)
DROP TABLE [OpeningToBoundarySurface]
GO
```

Figure C-2 (Continued).

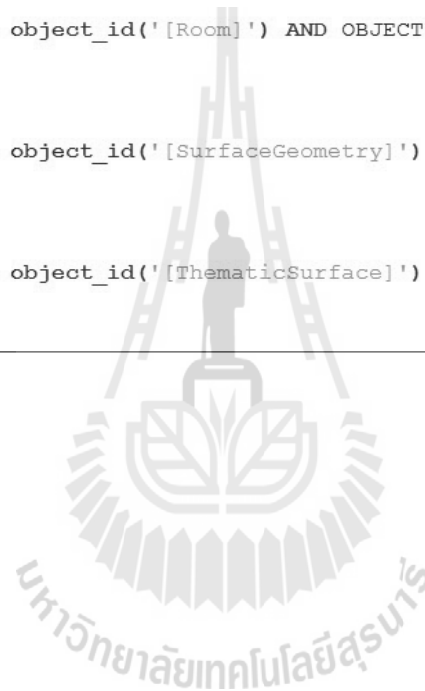
```
IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[RoofType]') AND OBJECTPROPERTY(id, 'IsUserTable') = 1)
DROP TABLE [RoofType]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[Room]') AND OBJECTPROPERTY(id, 'IsUserTable') = 1)
DROP TABLE [Room]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[SurfaceGeometry]') AND OBJECTPROPERTY(id, 'IsUserTable') = 1)
DROP TABLE [SurfaceGeometry]
GO

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = object_id('[ThematicSurface]') AND OBJECTPROPERTY(id, 'IsUserTable') = 1)
DROP TABLE [ThematicSurface]
GO
```

Figure C-2 (Continued).



```
CREATE TABLE [AccessoryUnit]
(
    [ID] int NOT NULL IDENTITY (1, 1) NOT FOR REPLICATION,
    [accessoryUnitType] tinyint NULL,
    [buildingID] int NULL,
    [area] real NULL,
    [areaUnit] varchar(50) NULL,
    [volume] real NULL,
    [volumeUnit] varchar(50) NULL,
    [apartmentUnitID] int NULL
)
GO

CREATE TABLE [AccessoryUnitType]
(
    [ID] tinyint NOT NULL,
    [name] varchar(50) NULL,
    [description] varchar(50) NULL
)
GO

CREATE TABLE [ApartmentUnit]
(
    [ID] int NOT NULL IDENTITY (1, 1) NOT FOR REPLICATION,
    [unitNo] varchar(10) NOT NULL,
    [roomID] int NULL,
    [roomCount] tinyint NULL,
    [floorNo] tinyint NULL,
    [area] real NULL,
    [areaUnit] varchar(1000) NULL,
    [volume] real NULL,
    [volumeUnit] varchar(1000) NULL
)
GO
```

Figure C-3 Create tables.


```

CREATE TABLE [Building]
(
  [ID] int NOT NULL,
  [buildingNo] varchar(10) NOT NULL,
  [plotNo] varchar(256) NOT NULL,
  [buildingParentID] int NULL,
  [buildingRootID] int NULL,
  [buildingType] tinyint NULL,    -- SIG3D: Classification of the actual usage of Building or BuildingPart as given by the relevant national regulations, information communities
  or specific applications.
  [roofType] tinyint NULL,    -- bSI: Basic configuration of the roof in terms of the different roof shapes
  [roofTypeCodespace] varchar(4000) NULL,
  [yearOfConstruction] date NULL,    -- SIG3D: Year of completion of this Building or BuildingPart.
  [yearOfDemolition] date NULL,    -- SIG3D: Year of demolition of this Building or BuildingPart.
  [measuredHeight] float NULL,    -- SIG3D: Measured or calculated distance between the highest point of the roof construction and the lowest point of the terrain intersection of
  the building
  [storeysAboveGround] int NULL,    -- SIG3D: Number of storeys mainly above ground
  [storeyHeightsAboveGround] varchar(4000) NULL,    -- SIG3D: List of heights for each storey above ground
  [storeyHeightAGUnit] varchar(4000) NULL,
  [storeysBelowGround] int NULL,    -- SIG3D: Number of storeys mainly below ground
  [storeyHeightsBelowGround] varchar(4000) NULL,    -- SIG3D: List of heights for each storey below ground
  [storeyHeightBGUnit] varchar(4000) NULL,
  [lod0FootPrintID] int NULL,    -- SIG3D: Relation to a LOD0 foot print geometry of Building or BuildingPart. The LOD concept for buildings or building parts is defined in
  chapter ...
  [lod1MultiSurfaceID] int NULL,    -- SIG3D: Relation to a LOD1 surface geometry of Building or BuildingPart. The LOD concept for buildings or building parts is defined in
  chapter ... Geometrically, the LOD1 solid geometry and surface geometry are identical.
  [lod1SolidID] int NULL,    -- SIG3D: Relation to a LOD1 solid geometry of Building or BuildingPart. The LOD concept for buildings or building parts is defined in chapter ...
  [lod4MultiSurfaceID] int NULL,    -- SIG3D: Relation to a LOD4 surface geometry of Building or BuildingPart. The LOD concept for buildings or building parts is defined in
  chapter ... A LOD4 surface geometry may be used in addition to a LOD4 solid geometry (e.g. to model roof overhangs), or it may replace a LOD 4 solid geometry (e.g. buildings
  without ground plate).
  [lod4SolidID] int NULL    -- SIG3D: Relation to a LOD4 solid geometry of Building or BuildingPart. The LOD concept for buildings or building parts is defined in chapter ...
)
GO

```

Figure C-3 (Continued).

```

CREATE TABLE [BuildingInstallation]
(
  [ID] int NOT NULL IDENTITY (1, 1) NOT FOR REPLICATION,
  [objectClassID] tinyint NOT NULL,
  [buildingID] int NULL,
  [roomID] int NULL,
  [lod4BREPID] int NULL,
  [lod4ImplicitRefPoint] geometry NULL,
  [lod4OtherGeom] geometry NULL, -- SIG3D: LOD4 geometry of BuildingInstallation
  [lod4ImplicitRepID] int NULL,
  [lod4ImplicitTransformation] varchar(1000) NULL
)
GO

CREATE TABLE [BuildingType]
(
  [ID] tinyint NOT NULL,
  [name] varchar(50) NULL,
  [description] varchar(50) NULL
)
GO

CREATE TABLE [CityModel]
(
  [ID] int NOT NULL IDENTITY (1, 1) NOT FOR REPLICATION,
  [GMLID] varchar(256) NULL,
  [cityModelType] tinyint NULL,
  [description] varchar(4000) NULL,
  [envelope] geometry NULL,
  [creationDate] datetimeoffset(7) NULL,
  [terminationDate] datetimeoffset(7) NULL,
  [lastModificationDate] datetimeoffset(7) NULL,
  [updatingPerson] varchar(256) NULL,
  [reasonForUpdate] varchar(4000) NULL,
  [lineage] varchar(256) NULL
)
GO

```

Figure C-3 (Continued).

```

CREATE TABLE [CityModelType]
(
  [ID] tinyint NOT NULL,
  [name] varchar(50) NULL,
  [description] varchar(50) NULL
)
GO

CREATE TABLE [CityObject]
(
  [ID] int NOT NULL IDENTITY (1, 1) NOT FOR REPLICATION,
  [objectClassID] tinyint NOT NULL,
  [GMLID] varchar(256) NULL,
  [terminationDate] datetimeoffset(7) NULL, -- ???
  [relativeToTerrain] varchar(256) NULL, -- Location of the city object relative to the surrounding terrain.
  [relativeToWater] varchar(256) NULL, -- Location of the city object relative to the surrounding water surface.
  [envelope] geometry NULL,
  [lastModificationDate] datetimeoffset(7) NULL,
  [updatingPerson] varchar(256) NULL,
  [reasonForUpdate] varchar(4000) NULL,
  [lineage] varchar(256) NULL,
  [xmlSource] text NULL
)
GO

CREATE TABLE [CityObjectGroup]
(
  [ID] int NOT NULL IDENTITY (1, 1) NOT FOR REPLICATION,
  [parentCityObjectID] int NULL,
  [BREPID] int NULL,
  [otherGeom] geometry NULL
)
GO

CREATE TABLE [CityObjectMember]
(
  [cityModelID] int NOT NULL,
  [cityObjectID] int NOT NULL
)
GO

```

Figure C-3 (Continued).

```
CREATE TABLE [CommonPropertyUnit]
(
    [ID] int NOT NULL IDENTITY (1, 1) NOT FOR REPLICATION,
    [commonPropertyType] tinyint NULL,
    [area] real NULL,
    [areaUnit] varchar(1000) NULL,
    [volume] real NULL,
    [volumeUnit] varchar(1000) NULL,
    [apartmentUnitID] int NULL,
    [roomID] int NULL,
    [buildingInstallationID] int NULL
)
GO

CREATE TABLE [CommonPropertyUnitType]
(
    [ID] tinyint NOT NULL,
    [name] varchar(50) NULL,
    [description] varchar(50) NULL
)
GO

CREATE TABLE [FloorNo]
(
    [ID] tinyint NOT NULL,
    [name] varchar(50) NULL,
    [description] varchar(50) NULL
)
GO

CREATE TABLE [GroupToCityObject]
(
    [cityObjectID] int NOT NULL,
    [cityObjectGroupID] int NOT NULL,
    [role] varchar(1000) NULL
)
GO
```

Figure C-3 (Continued).

```

CREATE TABLE [ImplicitGeometry]
(
  [ID] int NOT NULL IDENTITY (1, 1) NOT FOR REPLICATION,
  [mimeType] varchar(256) NULL,    -- Mime type of the referenced external geometric object (attribute libraryObject).
  [referenceToLibrary] varchar(4000) NULL,    -- Base point of the object in the world coordinate system.
  [libraryObject] text NULL,    -- External link to a prototype geometry.
  [relativeBREPID] int NULL,    -- Geometry of the prototype, specified in a local coordinate system.
  [relativeOtherGeom] geometry NULL    -- Mathematical transformation (translation, rotation and scaling) between the prototype geometry and the actual spatial position of the
  object.
)
GO

CREATE TABLE [LegalPropertyType]
(
  [ID] tinyint NOT NULL,
  [name] varchar(50) NULL,
  [description] varchar(50) NULL
)
GO

CREATE TABLE [ObjectClass]
(
  [ID] tinyint NOT NULL,
  [className] varchar(256) NOT NULL,
  [superClassID] tinyint NULL
)
GO

CREATE TABLE [Opening]
(
  [ID] int NOT NULL IDENTITY (1, 1) NOT FOR REPLICATION,
  [objectClassID] tinyint NOT NULL,
  [lod4MultiSurfaceID] int NULL,    -- SIG3D: LOD4 surface geometry of Door or Window
  [lod4ImplicitRepID] int NULL,
  [lod4ImplicitRefPoint] geometry NULL,
  [lod4ImplicitTransformation] varchar(1000) NULL
)
GO

```

Figure C-3 (Continued).

```

CREATE TABLE [OpeningToBoundarySurface]
(
    [openingID] int NOT NULL,
    [boundarySurfaceID] int NOT NULL
)
GO

CREATE TABLE [RoofType]
(
    [ID] tinyint NOT NULL,
    [name] varchar(50) NULL,
    [description] varchar(50) NULL
)
GO

CREATE TABLE [Room]
(
    [ID] int NOT NULL IDENTITY (1, 1) NOT FOR REPLICATION,
    [buildingID] int NOT NULL,
    [lod4MultiSurfaceID] int NULL,    -- SIG3D: Relation to aLOD4 surface geometry of Room. The LOD concept for buildings or building parts is defined in chapter ...
    [lod4SolidID] int NULL,         -- SIG3D: Relation to a LOD4 solid geometry of Room. The LOD concept for buildings or building parts is defined in chapter ...
    [legalPropertyType] tinyint NULL
)
GO

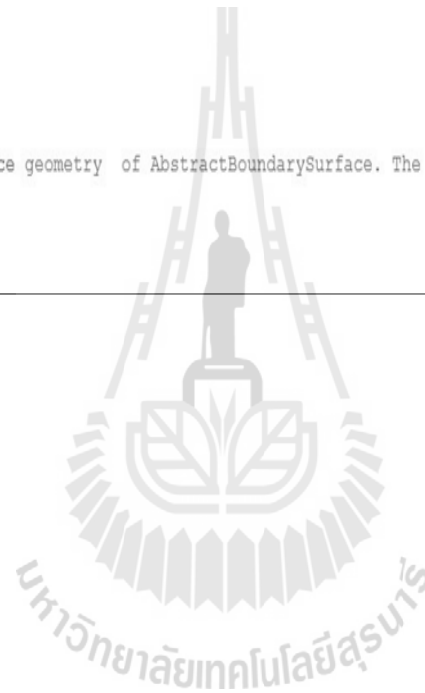
CREATE TABLE [SurfaceGeometry]
(
    [ID] int NOT NULL,
    [GMLID] varchar(256) NULL,
    [parentID] int NULL,
    [rootID] int NULL,
    [isSolid] binary(10) NULL,
    [isComposite] binary(10) NULL,
    [isXLink] binary(10) NULL,
    [geometry] geometry NULL,
    [solidGeometry] geometry NULL,
    [implicitGeometry] geometry NULL,
    [cityObjectID] int NULL
)
GO

```

Figure C-3 (Continued).

```
CREATE TABLE [ThematicSurface]
(
  [ID] int NOT NULL IDENTITY (1, 1) NOT FOR REPLICATION,
  [objectClassID] tinyint NOT NULL,
  [roomID] int NULL,
  [buildingID] int NULL,
  [buildingInstallationID] int NULL,
  [lod4MultiSurfaceID] int NULL -- SIG3D: Relation to a LOD4 surface geometry of AbstractBoundarySurface. The LOD concept for boundary surfaces of buildings or building parts
  is defined in chapter ...
)
GO
```

Figure C-3 (Continued).



```
ALTER TABLE [AccessoryUnit]
  ADD CONSTRAINT [PK_AccessoryUnit]
    PRIMARY KEY CLUSTERED ([ID])
GO

ALTER TABLE [AccessoryUnitType]
  ADD CONSTRAINT [PK_AccessoryUnitType]
    PRIMARY KEY CLUSTERED ([ID])
GO

ALTER TABLE [ApartmentUnit]
  ADD CONSTRAINT [PK_ApartmentUnit]
    PRIMARY KEY CLUSTERED ([ID])
GO

ALTER TABLE [ApartmentUnit]
  ADD CONSTRAINT [Unique_ApartmentUnit] UNIQUE NONCLUSTERED ([unitNo])
GO

ALTER TABLE [Building]
  ADD CONSTRAINT [PK_Building]
    PRIMARY KEY CLUSTERED ([ID])
GO

ALTER TABLE [Building]
  ADD CONSTRAINT [Unique_BuildingNo] UNIQUE NONCLUSTERED ([buildingNo])
GO

ALTER TABLE [BuildingInstallation]
  ADD CONSTRAINT [PK_BuildingInstallation]
    PRIMARY KEY CLUSTERED ([ID])
GO

ALTER TABLE [BuildingType]
  ADD CONSTRAINT [PK_BuildingType]
    PRIMARY KEY CLUSTERED ([ID])
GO
```

Figure C-4 Create primary keys (PK), indexes, unique, checks.


```
ALTER TABLE [CityModel]
  ADD CONSTRAINT [PK_CityModel]
    PRIMARY KEY CLUSTERED ([ID])
GO

ALTER TABLE [CityModelType]
  ADD CONSTRAINT [PK_CityModelType]
    PRIMARY KEY CLUSTERED ([ID])
GO

ALTER TABLE [CityObject]
  ADD CONSTRAINT [PK_CityObject]
    PRIMARY KEY CLUSTERED ([ID])
GO

ALTER TABLE [CityObjectGroup]
  ADD CONSTRAINT [PK_CityObjectGroup]
    PRIMARY KEY CLUSTERED ([ID])
GO

ALTER TABLE [CityObjectMember]
  ADD CONSTRAINT [PK_CityObjectMember]
    PRIMARY KEY CLUSTERED ([cityModelID],[cityObjectID])
GO

ALTER TABLE [CommonPropertyUnitType]
  ADD CONSTRAINT [PK_CommonPropertyUnitType]
    PRIMARY KEY CLUSTERED ([ID])
GO

ALTER TABLE [FloorNo]
  ADD CONSTRAINT [PK_FloorNo]
    PRIMARY KEY CLUSTERED ([ID])
GO

ALTER TABLE [GroupToCityObject]
  ADD CONSTRAINT [PK_GroupToCityObject]
    PRIMARY KEY CLUSTERED ([cityObjectGroupID],[cityObjectID])
GO
```

Figure C-4 (Continued).

```
ALTER TABLE [ImplicitGeometry]
  ADD CONSTRAINT [PK_ImplicitGeometry]
    PRIMARY KEY CLUSTERED ([ID])
GO

ALTER TABLE [LegalPropertyType]
  ADD CONSTRAINT [PK_LegalPropertyType]
    PRIMARY KEY CLUSTERED ([ID])
GO

ALTER TABLE [ObjectClass]
  ADD CONSTRAINT [PK_ObjectClass]
    PRIMARY KEY CLUSTERED ([ID])
GO

ALTER TABLE [Opening]
  ADD CONSTRAINT [PK_Opening]
    PRIMARY KEY CLUSTERED ([ID])
GO

ALTER TABLE [OpeningToBoundarySurface]
  ADD CONSTRAINT [PK_OpeningToBoundarySurface]
    PRIMARY KEY CLUSTERED ([boundarySurfaceID],[openingID])
GO

ALTER TABLE [RoofType]
  ADD CONSTRAINT [PK_RoofType]
    PRIMARY KEY CLUSTERED ([ID])
GO

ALTER TABLE [Room]
  ADD CONSTRAINT [PK_Room]
    PRIMARY KEY CLUSTERED ([ID])
GO

ALTER TABLE [SurfaceGeometry]
  ADD CONSTRAINT [PK_SurfaceGeometry]
    PRIMARY KEY CLUSTERED ([ID])
GO
```

Figure C-4 (Continued).

```
ALTER TABLE [CityModel]
  ADD CONSTRAINT [PK_CityModel]
    PRIMARY KEY CLUSTERED ([ID])
GO

ALTER TABLE [CityModelType]
  ADD CONSTRAINT [PK_CityModelType]
    PRIMARY KEY CLUSTERED ([ID])
GO

ALTER TABLE [CityObject]
  ADD CONSTRAINT [PK_CityObject]
    PRIMARY KEY CLUSTERED ([ID])
GO

ALTER TABLE [CityObjectGroup]
  ADD CONSTRAINT [PK_CityObjectGroup]
    PRIMARY KEY CLUSTERED ([ID])
GO

ALTER TABLE [CityObjectMember]
  ADD CONSTRAINT [PK_CityObjectMember]
    PRIMARY KEY CLUSTERED ([cityModelID],[cityObjectID])
GO

ALTER TABLE [CommonPropertyUnitType]
  ADD CONSTRAINT [PK_CommonPropertyUnitType]
    PRIMARY KEY CLUSTERED ([ID])
GO

ALTER TABLE [FloorNo]
  ADD CONSTRAINT [PK_FloorNo]
    PRIMARY KEY CLUSTERED ([ID])
GO

ALTER TABLE [GroupToCityObject]
  ADD CONSTRAINT [PK_GroupToCityObject]
    PRIMARY KEY CLUSTERED ([cityObjectGroupID],[cityObjectID])
GO
```

Figure C-4 (Continued).

```
CREATE INDEX [surface_geom_inx]
  ON [SurfaceGeometry] ([GMLID] ASC)
  WITH FILLFACTOR = 90
GO

CREATE INDEX [surface_geom_parent_fkx]
  ON [SurfaceGeometry] ([parentID] ASC)
  WITH FILLFACTOR = 90
GO

CREATE INDEX [surface_geom_root_fkx]
  ON [SurfaceGeometry] ([rootID] ASC)
  WITH FILLFACTOR = 90
GO

CREATE INDEX [surface_geom_cityobj_fkx]
  ON [SurfaceGeometry] ([cityObjectID] ASC)
  WITH FILLFACTOR = 90
GO

ALTER TABLE [ThematicSurface]
  ADD CONSTRAINT [PK_ThematicSurface]
    PRIMARY KEY CLUSTERED ([ID])
GO
```

Figure C-4 (Continued).

```

ALTER TABLE [AccessoryUnit] ADD CONSTRAINT [FK_AccessoryUnit_ApartmentUnit]
    FOREIGN KEY ([apartmentUnitID]) REFERENCES [ApartmentUnit] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [AccessoryUnit] ADD CONSTRAINT [FK_AccessoryUnit_Building]
    FOREIGN KEY ([buildingID]) REFERENCES [Building] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [ApartmentUnit] ADD CONSTRAINT [FK_ApartmentUnit_Room]
    FOREIGN KEY ([roomID]) REFERENCES [Room] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [Building] ADD CONSTRAINT [FK_Building_Building]
    FOREIGN KEY ([buildingParentID]) REFERENCES [Building] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [Building] ADD CONSTRAINT [FK_Building_Building_02]
    FOREIGN KEY ([buildingRootID]) REFERENCES [Building] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [Building] ADD CONSTRAINT [FK_Building_CityObject]
    FOREIGN KEY ([ID]) REFERENCES [CityObject] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [Building] ADD CONSTRAINT [FK_Building_SurfaceGeometry]
    FOREIGN KEY ([lod0FootPrintID]) REFERENCES [SurfaceGeometry] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [Building] ADD CONSTRAINT [FK_Building_SurfaceGeometry_02]
    FOREIGN KEY ([lod1MultiSurfaceID]) REFERENCES [SurfaceGeometry] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [Building] ADD CONSTRAINT [FK_Building_SurfaceGeometry_03]
    FOREIGN KEY ([lod1SolidID]) REFERENCES [SurfaceGeometry] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [Building] ADD CONSTRAINT [FK_Building_SurfaceGeometry_04]
    FOREIGN KEY ([lod4MultiSurfaceID]) REFERENCES [SurfaceGeometry] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

```

Figure C-5 Create foreign key (FK) constraints.

```
ALTER TABLE [AccessoryUnit] ADD CONSTRAINT [FK_AccessoryUnit_ApartmentUnit]
    FOREIGN KEY ([apartmentUnitID]) REFERENCES [ApartmentUnit] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [AccessoryUnit] ADD CONSTRAINT [FK_AccessoryUnit_Building]
    FOREIGN KEY ([buildingID]) REFERENCES [Building] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [ApartmentUnit] ADD CONSTRAINT [FK_ApartmentUnit_Room]
    FOREIGN KEY ([roomID]) REFERENCES [Room] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [Building] ADD CONSTRAINT [FK_Building_Building]
    FOREIGN KEY ([buildingParentID]) REFERENCES [Building] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [Building] ADD CONSTRAINT [FK_Building_Building_02]
    FOREIGN KEY ([buildingRootID]) REFERENCES [Building] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [Building] ADD CONSTRAINT [FK_Building_CityObject]
    FOREIGN KEY ([ID]) REFERENCES [CityObject] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [Building] ADD CONSTRAINT [FK_Building_SurfaceGeometry]
    FOREIGN KEY ([lod0FootPrintID]) REFERENCES [SurfaceGeometry] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [Building] ADD CONSTRAINT [FK_Building_SurfaceGeometry_02]
    FOREIGN KEY ([lod1MultiSurfaceID]) REFERENCES [SurfaceGeometry] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [Building] ADD CONSTRAINT [FK_Building_SurfaceGeometry_03]
    FOREIGN KEY ([lod1SolidID]) REFERENCES [SurfaceGeometry] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [Building] ADD CONSTRAINT [FK_Building_SurfaceGeometry_04]
    FOREIGN KEY ([lod4MultiSurfaceID]) REFERENCES [SurfaceGeometry] ([ID]) ON DELETE No Action ON UPDATE No Action
GO
```

Figure C-5 (Continued).

```
ALTER TABLE [CityObjectGroup] ADD CONSTRAINT [FK_CityObjectGroup_CityObject]
    FOREIGN KEY ([ID]) REFERENCES [CityObject] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [CityObjectMember] ADD CONSTRAINT [FK_CityObjectMember_CityModel]
    FOREIGN KEY ([cityModelID]) REFERENCES [CityModel] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [CityObjectMember] ADD CONSTRAINT [FK_CityObjectMember_CityObject]
    FOREIGN KEY ([cityObjectID]) REFERENCES [CityObject] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [CommonPropertyUnit] ADD CONSTRAINT [FK_OtherPropertyUnit_ApartmentUnit]
    FOREIGN KEY ([apartmentUnitID]) REFERENCES [ApartmentUnit] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [CommonPropertyUnit] ADD CONSTRAINT [FK_OtherPropertyUnit_BuildingInstallation]
    FOREIGN KEY ([buildingInstallationID]) REFERENCES [BuildingInstallation] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [CommonPropertyUnit] ADD CONSTRAINT [FK_OtherPropertyUnit_Room]
    FOREIGN KEY ([roomID]) REFERENCES [Room] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [GroupToCityObject] ADD CONSTRAINT [FK_GroupToCityObject_CityObject]
    FOREIGN KEY ([cityObjectID]) REFERENCES [CityObject] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [GroupToCityObject] ADD CONSTRAINT [FK_GroupToCityObject_CityObjectGroup]
    FOREIGN KEY ([cityObjectGroupID]) REFERENCES [CityObjectGroup] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [ImplicitGeometry] ADD CONSTRAINT [FK_ImplicitGeometry_SurfaceGeometry]
    FOREIGN KEY ([relativeBREPID]) REFERENCES [SurfaceGeometry] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [ObjectClass] ADD CONSTRAINT [FK_ObjectClass_ObjectClass]
    FOREIGN KEY ([superClassID]) REFERENCES [ObjectClass] ([ID]) ON DELETE No Action ON UPDATE No Action
GO
```

Figure C-5 (Continued).

```

ALTER TABLE [Opening] ADD CONSTRAINT [FK_Opening_CityObject]
    FOREIGN KEY ([ID]) REFERENCES [CityObject] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [Opening] ADD CONSTRAINT [FK_Opening_ImplicitGeometry]
    FOREIGN KEY ([lod4ImplicitRepID]) REFERENCES [ImplicitGeometry] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [Opening] ADD CONSTRAINT [FK_Opening_ObjectClass]
    FOREIGN KEY ([objectClassID]) REFERENCES [ObjectClass] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [Opening] ADD CONSTRAINT [FK_Opening_SurfaceGeometry]
    FOREIGN KEY ([lod4MultiSurfaceID]) REFERENCES [SurfaceGeometry] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [OpeningToBoundarySurface] ADD CONSTRAINT [FK_OpeningToBoundarySurface_BoundarySurface]
    FOREIGN KEY ([boundarySurfaceID]) REFERENCES [ThematicSurface] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [OpeningToBoundarySurface] ADD CONSTRAINT [FK_OpeningToBoundarySurface_Opening]
    FOREIGN KEY ([openingID]) REFERENCES [Opening] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [Room] ADD CONSTRAINT [FK_Room_SurfaceGeometry]
    FOREIGN KEY ([lod4MultiSurfaceID]) REFERENCES [SurfaceGeometry] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [Room] ADD CONSTRAINT [FK_Room_SurfaceGeometry_02]
    FOREIGN KEY ([lod4SolidID]) REFERENCES [SurfaceGeometry] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [Room] ADD CONSTRAINT [FK_Room_Building]
    FOREIGN KEY ([buildingID]) REFERENCES [Building] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [Room] ADD CONSTRAINT [FK_Room_CityObject]
    FOREIGN KEY ([ID]) REFERENCES [CityObject] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

```

Figure C-5 (Continued).


```

ALTER TABLE [SurfaceGeometry] ADD CONSTRAINT [FK_SurfaceGeometry_CityObject]
    FOREIGN KEY ([cityObjectID]) REFERENCES [CityObject] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [SurfaceGeometry] ADD CONSTRAINT [FK_SurfaceGeometry_SurfaceGeometry]
    FOREIGN KEY ([parentID]) REFERENCES [SurfaceGeometry] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [SurfaceGeometry] ADD CONSTRAINT [FK_SurfaceGeometry_SurfaceGeometry_02]
    FOREIGN KEY ([rootID]) REFERENCES [SurfaceGeometry] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [ThematicSurface] ADD CONSTRAINT [FK_ThematicSurface_Building]
    FOREIGN KEY ([buildingID]) REFERENCES [Building] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [ThematicSurface] ADD CONSTRAINT [FK_ThematicSurface_BuildingInstallation]
    FOREIGN KEY ([buildingInstallationID]) REFERENCES [BuildingInstallation] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [ThematicSurface] ADD CONSTRAINT [FK_ThematicSurface_ObjectClass]
    FOREIGN KEY ([objectClassID]) REFERENCES [ObjectClass] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [ThematicSurface] ADD CONSTRAINT [FK_ThematicSurface_SurfaceGeometry]
    FOREIGN KEY ([lod4MultiSurfaceID]) REFERENCES [SurfaceGeometry] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [ThematicSurface] ADD CONSTRAINT [FK_ThematicSurface_Room]
    FOREIGN KEY ([roomID]) REFERENCES [Room] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

ALTER TABLE [ThematicSurface] ADD CONSTRAINT [FK_ThematicSurface_CityObject]
    FOREIGN KEY ([ID]) REFERENCES [CityObject] ([ID]) ON DELETE No Action ON UPDATE No Action
GO

```

Figure C-5 (Continued).

```

DELETE FROM ObjectClass;

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (0,'Undefined',NULL);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (1,'_GML',NULL);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (2,'_Feature',1);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (3,'_CityObject',2);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (4,'_Site',3);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (5,'CityObjectGroup',3);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (6,'_AbstractBuilding',4);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (7,'BuildingPart',6);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (8,'Building',6);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (9,'BuildingInstallation',3);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (10,'_BuildingBoundarySurface',3);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (11,'BuildingCeilingSurface',10);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (12,'InteriorBuildingWallSurface',10);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (13,'BuildingFloorSurface',10);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (14,'BuildingRoofSurface',10);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (15,'BuildingWallSurface',10);

```

Figure C-6 Object class instances.

```

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (16,'BuildingGroundSurface',10);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (17,'BuildingClosureSurface',10);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (18,'_BuildingOpening',3);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (19,'BuildingWindow',18);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (20,'BuildingDoor',18);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (21,'BuildingRoom',3);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (22,'FeatureCollection',2);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (23,'CityModel',22);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (24,'OuterBuildingCeilingSurface',10);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (25,'OuterBuildingFloorSurface',10);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (26,'_SpatialUnit',3);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (27,'_LegalSpaceBuildingUnit',26);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (28,'ApartmentUnit',27);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (29,'CommonPropertyUnit',27);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (30,'AccessoryUnit',27);

INSERT INTO ObjectClass ( ID , className , superClassID )
VALUES (31,'LegalSpaceBuilding',27);

```

Figure C-6 (Continued).

```

CREATE PROCEDURE [dbo].[sp_generate_unique_App_id]
AS
DECLARE @ID          AS INT,
        @IdSeed      AS INT,
        @EOFFTable   AS BIT

DECLARE curTable CURSOR FOR SELECT ID FROM ApartmentUnit

OPEN curTable
SELECT @EOFFTable = 1
BEGIN TRANSACTION
---Get the first record and check for EOF
FETCH curTable INTO @ID
IF @@ERROR <> 0 GOTO ERROR_HANDLER
IF @@FETCH_STATUS <> 0
BEGIN
    SELECT @EOFFTable = 0
END
WHILE NOT (@EOFFTable = 0)
BEGIN
    SELECT @IdSeed = Seed FROM counter WHERE Unit = 'AP'
    UPDATE ApartmentUnit SET unitNo = 'UNIT-' + LTRIM(@IdSeed + 1) WHERE ID = @ID
    UPDATE Counter SET Seed = @IdSeed + 1 WHERE Unit = 'AP'

    FETCH curTable INTO @ID
    IF @@FETCH_STATUS <> 0
        BEGIN
            SELECT @EOFFTable = 0
        END
    END
END
COMMIT TRANSACTION
CLOSE curTable
DEALLOCATE curTable
SELECT 'Unique ID updated successfully !!!'
RETURN
ERROR_HANDLER:
    ROLLBACK TRANSACTION
    RAISERROR ('Could not update Unique ID.',16,1)
    CLOSE curTable
    DEALLOCATE curTable

```

Figure C-7 Store procedure for generation of unique identifier for apartment unit.

CURRICULUM VITAE

Name: Tashi

Date of Birth: 05 April, 1979

Place of Birth: Punakha, Bhutan.

Education:

2000 Bachelor of Science. Sherubtse College (Affiliated to Delhi University, India), Tashigang, Eastern Bhutan.

2006 Post Graduation Diploma in Surveying and Mapping (Course 500 Survey Engineering). Indian Institute of Surveying and Mapping (IIMS), Survey of India. Uppal, Hyderabad, India.

Publications:

Ongsomwang, S., and Tashi. (2015). 3D Building cadastral design using CityGML: A Case study of Thimphu City, Bhutan. Journal of Remote Sensing and GIS Association of Thailand. Vol. 17 (1). (Accepted on 21 January 2016).

Grants and Fellowships:

Government of India scholarship.

Thailand International Cooperation Agency scholarship.

Position and Place of Work:

Sr. Survey Engineer under Urban Planning Division, Gelephu Thromde, Sarpang, Bhutan -00975.