Evaluations on XML Standards for Actual Applications

by

Jiemin Zhang

B.Eng., Jilin University, 2006

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

The Faculty of Graduate Studies

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

September, 2008

© Jiemin Zhang 2008
Abstract

XML is a popular data format today, particular for data exchange over the web or among applications. Standards based on XML are widely used in lots of areas and for a number of applications. However, XML standards are still in their early stage therefore it is interesting to explore how they actually work in reality. In this thesis, we evaluate XML standards when using for actual applications, and propose a set of hypotheses concluding both advantages and challenges of using XML in real applications. We verify our hypotheses through a case study on a civil engineering application, and also review a few other areas to support them in general. Our study shows that XML standards have powerful strengths that make XML an important medium to support data exchange, but hard to use and far from perfect.
Table of Contents

Abstract ........................................... ii
Table of Contents ................................... iii
List of Tables ....................................... v
List of Figures ...................................... vi
Acknowledgements .................................... vii

I Thesis ............................................. 1

1 Introduction ...................................... 2
1.1 Motivations ..................................... 2
1.2 Contributions ................................... 3
1.3 Hypotheses ...................................... 3

2 Background ....................................... 6
2.1 The ARTIFACT project ......................... 6
2.1.1 Motivations .................................. 6
2.1.2 Overall architecture and design feature analysis ..... 8
2.2 3D CAD model and Autodesk Revit Architecture ...... 9
2.3 XML and XML standards ....................... 9
2.3.1 XML ......................................... 9
2.3.2 XML standards ............................ 11

3 A case study – Developing an application for civil engineering with XML standards ........ 12
3.1 Goals and motivations of the application ........ 12
3.2 The feature ontology ........................... 14
3.3 Candidate standards ........................... 18
3.3.1 Relational database (Microsoft Access) ........ 18
Table of Contents

3.3.2 XML based standards ........................................ 20
3.3.3 Comparisons ............................................. 26
3.3.4 Our decision—choose ifcXML ............................ 28
3.4 Using ifcXML for the application ............................ 29
3.4.1 How to extract ifcXML files from Revit ................. 29
3.4.2 Observations and analysis on ifcXML: complexity, content, and challenges .......................... 29
3.4.3 Querying processing ...................................... 36
3.5 The application .............................................. 41
  3.5.1 Overview of querying process ........................... 41
  3.5.2 Limitations and current version ......................... 43
  3.5.3 Languages and technologies ............................ 43
  3.5.4 User Interfaces .......................................... 45
3.6 Case study conclusion - XML standards for civil engineering .................................. 48

4 A review on XML standards for E-business .................. 53
  4.1 Motivations and paper summaries ........................ 53
  4.2 Synthesis - XML standards for E-business ............... 55

5 More General Discussions ...................................... 61
  5.1 Reviews on other industries .............................. 61
    5.1.1 Introduction to selected papers ..................... 61
    5.1.2 Synthesis ........................................... 62
  5.2 A summary ................................................. 70

6 Conclusion and Future Work ................................... 72

Bibliography .................................................. 74
List of Tables

3.1 Portion of the feature ontology ......................... 15
3.2 A comparison on how the four selected standards – Access, gbXML, DWF (stands for DWF-content XML) and ifcXML – support the feature ontology ....................... 26
3.3 What is provided and what is derived in ifcXML – for Wall component .......................... 31
5.1 Summary chart ............................................ 71
List of Figures

2.1 Building design involves a variety of groups and documents dealing with not well communicated or managed implicit knowledge. Figure taking from [29] ....................... 7
2.2 The Overall Architecture of ARTIFACT. Figure taking from [15] .................. 8
2.3 Revit MEP 2008 with a 3D View of a Building Model ..... 10
2.4 XML example with its tree view .................. 11

3.1 Floor and properties views provided in Revit .......... 13
3.2 Access sample data from model “Single_Room_With_Beam” . 19
3.3 Portion of tree view of exported gbXML file from model “Single_Room_With_Beam” ........................................ 21
3.4 Tree view of “Wall” category in DWF-content XML ........ 23
3.5 Portion of DWF-content XML data of a wall instance in model “Single_Room_With_Beam” .......................... 24
3.6 Hierarchical representation of exported ifcXML file ........ 25
3.7 A 3D wall object with corresponding ifcXML representations 30
3.8 Properties and relationships with IdRef paths showing their linkage to a wall object in ifcXML .......................... 31
3.9 Linking “Door Opening” to a Wall in ifcXML ............... 32
3.10 IFC Engine Viewer ........................................ 36
3.11 Linking property "IsExternal" to a Wall in ifcXML ........ 38
3.12 Application querying process. Figure taking from [23] .... 42
3.13 A portion of FBM-xml example ................................ 45
3.14 FBM browsing UI .................................. 46
3.15 Query from library UI .............................. 47
3.16 Property Filter UI ................................. 48
3.17 Quantification and Grouping UI .......................... 49
Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr. Rachel Pottinger. She has assisted me in numerous ways and provided me sound advice throughout my research and thesis-writing period. Without her encouragement, insight, and guidance, I would not have completed this thesis.

I would like to thank our research team for the ARTIFACT project: Dr. Sheryl Staub-French, Madhav Nepal, and Norman Petersen for providing expert knowledge on civil engineering and Michael Lawrence and April Webster who helped in reviewing and completing this thesis.

In addition, I am grateful to Dr. George Tsiknis for being my second reader and providing insight to make this thesis more complete. I am also thankful to all the members of the Data Management and Mining Lab. They provided a great atmosphere to learn as well as to have fun. Many thanks to Kan Cai, Rong Yi, Haijun Qiao and all my friends in both Vancouver and elsewhere for their support.

I wish to thank Mike Tsai for his constant love and encouragement. Lastly, and most importantly, I wish to thank my parents, on whose unquestioning love and support I have relied throughout my entire childhood and education. To them I dedicate this thesis.
Part I
Thesis
Chapter 1

Introduction

1.1 Motivations

As a data exchange and storage format, XML is becoming more and more important. It has been widely used in a large number of areas and applications, particularly for information interchange. Additionally, information interchange is becoming one of the most crucial issues within the data management area, as many applications and most services over the web rely heavily on it. Many standards based on XML have been developed in various areas, to support information exchange among systems. However, XML standards are still in their early stage and not mature in many aspects. While many XML standards have been created, and there has been substantial work in individual target domains to see how well the existing standards are working, there have been no studies showing broadly the current status of XML standards to find directions for improvement.

Initially we were developing a civil engineering application for the ARTIFACT project (see Section 2.1), dealing with data extraction, particularly using XML. Through our development, we saw that XML standards are powerful enough to support our application; however they are not perfect and we met a lot challenges during the process. Therefore, we evaluated how XML standards work for the civil engineering application, and made a set of hypotheses on it. In order to see if our hypotheses are supported in general, we then explored in a few other areas, including e-business, finance, biology and legislation, to review how XML standards work for them.

The following sections of this chapter list our contributions (Section 1.2) and hypotheses (Section 1.3). We provide some background knowledge in the next chapter (Chapter 2). We then introduce the case study of the civil engineering application in Chapter 3, followed by a review on e-business in Chapter 4. A general discussion with studies on a few more areas is given in Chapter 5, and we conclude our work in the end with a list of future works (Chapter 6).
1.2 Contributions

Our contributions in this work can be concluded as following:

- Compared and evaluated a set of standards for a civil engineering application.
- Implemented a civil engineering application using ifcXML.
- Evaluated how ifcXML and XML standards work with civil engineering applications.
- Proposed a set of hypotheses on XML standards for actual standards drawn from the case study.
- Verified the hypotheses generally through exploring in more areas.

1.3 Hypotheses

Here we list all hypotheses of our work. We will introduce how we derive and verify them in the remaining chapters.

1. **XML has high expressiveness and flexibility**: XML is more expressive and flexible comparing to relational database, or other data formats. Users can define their own tags and structures to meet their own requirements in XML. Thus, XML standards potentially have larger capability to carry and exchange complex information.

2. **The schema of XML standards is sometimes complex**: in order to store data in a more flexible and extensible way, XML always represents information in a much harder manner that usually needs expert knowledge to understand. The more information an XML standard tries to support, the more complex the schema is.

3. **XML is not efficient enough**: XML is text, therefore its raw size is large even it only carries a small amount of information. That leads to inefficiency when exchanging a large amount of data and infeasibility when time is highly considered. In addition, current querying technologies on XML are still immature and inefficient; querying on XML files is really time consuming. That limits its usage on data storage.
4. **XML standards improve the facility of data exchange among applications:** applications can easily exchange information as long as they can generate XML files under agreed XML standards, regardless how they create and manage their data internally.

5. **Usually more than one XML standards are available in one area:** in most areas that we have reviewed, there exist several XML standards available. In some areas there are just a few, while in others there are dozens. The relationships among those standards in a single area is not explicit: some of them are designed for the same usage (i.e., they are redundant or compete with one another); others are not (i.e., they complement one another).

6. **There is no all-purpose XML standard for each area:** although there are several standards available in one area, there is no standard meeting requirements of all applications, i.e., suitable to be used everywhere in this area. If such a standard existed, developers in this area could just take use of it for any application and situation, without working on evaluating/comparing/integrating various standards. Unfortunately, it is not the fact. The fact is:

7. **Differences among available XML standards have to be identified:** different standards have different targets. Decisions on choosing the right standards, if any, to adopt have to be made; as a result users have to understand differences among all available standards, including their targets, content and complexity.

8. **Identifying differences among XML standards is usually not easy:** users usually need to create a comparison framework/criteria first, based on their requirements, and then evaluate standards in accordance of it. Also, there are usually underlying differences that require deep understandings on standards to derive. Many papers, have made great efforts on evaluating and comparing standards in one area (which verifies that it is not easy), but they only deployed a few of them and often target a specific usage.

9. **Sometimes none of the available XML standards is suitable:** although there are several standards available, sometimes none of them is capable enough to meet all requirements of an application. In this situation, either improving an existing one, integrating several ones, or creating a new one could be an option. None of them is easy though as the schemas of standards are various and sometimes complex. To
1.3. Hypotheses

create a new standard would also cause problems for exchanging data with other applications.

10. Coordination on standards is necessary: since data exchange is related to at least two applications, an agreement on standards among all participated applications is important. It is necessary for developers to make sure the potential partners would use the same or a similar standard; otherwise the interpretation among applications will be difficult. It is more severe for XML standards which have high flexibility and variety, particularly in the case of adopting a standard which is not widely used by others.

11. Transformation between XML and relational databases is needed sometimes: although XML is popular for exchanging data, there are still a large part of applications not storing data in XML format (most of them still choose relational databases for data storage; partly because of reasons in hypothesis 3), but only transmitting data through XML. In such cases, a mechanism to manage the transferring process between XML and data stored in another format is necessary.
Chapter 2

Background

In this chapter, we will provide some background information about the ARTIFACT project and main concepts that are mentioned in this work.

2.1 The ARTIFACT project

The application we will introduce in next chapter is a part of work of the ARTIFACT project. Here we give an introduction to this project.

ARTIFACT (Advanced Research, Techniques, and Informatics for Future Advantages in Construction Technology) is a project focusing on data management in Architecture, Engineering and Construction (AEC). More specifically, the goal of ARTIFACT is to develop novel technology for extracting, integrating and managing information in large building projects which is scattered in various applications and databases, in order to support more effective group collaboration in such projects.

2.1.1 Motivations

There are several motivations for ARTIFACT: In a large building project, a large number of individuals as well as various kinds of information will be involved. Those people, including “architects, engineering consultants, construction managers, facility maintenance organizations, and facility users” ([29]), have to collaborate to make sure decisions made during the project satisfy the requirements from all of them. On the other hand, estimators have to consider several kinds of information, including design models, cost estimates, schedules, etc. to make decisions. Meanwhile, some other kinds of documents, such as contracts, meeting minutes, specifications also need to be managed. Figure 2.1([29]) shows how problems in building design related to different groups and a variety of information.

For example, consider the question “what if I change the floor-to-floor height of the building”. The change on floor-to-floor height would not only result in revisions in the design, but may also lead to schedule update and
2.1. The ARTIFACT project

Figure 2.1: Building design involves a variety of groups and documents dealing with not well communicated or managed implicit knowledge. Figure taking from [29]

labors and materials cost change. Estimators have to consider all those consequences to decide if such a change should be made.

Moreover, the information in a building project is usually stored in different applications with different formats. Figure 2.2([15]) shows that 3D design information may be stored in Autodesk Revit with several exporting formats such as DWF, XML and relational databases (Microsoft Access); scheduling data could be created in Microsoft Project; construction photos with image formats and other unstructured documents may be scattered in various folders or even different computers and work sites.

Effective ways to interact and manage them is critical otherwise it would lead to inefficiencies to make decisions in the design process. Current technologies support people to create and manage a single kind of information separately, such as creating and save design models in Autodesk Revit and managing schedule data in Microsoft Access. However, they do not provide an automatic way to interact with all kinds of data; people usually have to relate different types of information manually. As observed in [8], "the ma-
The majority of time in design development meetings is spent on descriptive (35%) and explanatory (42%) tasks and very little time is spent on evaluative (12%) and predictive (11%) tasks. This is not efficient and would be very costly.

Therefore, people in ARTIFACT are trying to develop information management system providing ways to relate, interact, visualize all information in AEC, therefore to improve the efficiency in such projects.

2.1.2 Overall architecture and design feature analysis

Figure 2.2([15]) proposed the overall architecture of the system. As it shows, every kind of data, including scheduling, design, photos and cost stored in different applications, has several export mechanisms exporting to different databases or formats. Through those export files, we analyze each kind of data and coordinate all of them, then provide a unified browser or interfaces to support information navigation and querying.

The ARTIFACT project is still in its early stages and only limited analysis in each kind of data has been done. The work related to this thesis and the application introduced in the next chapter focus only on design information part which is highlighted in Figure 2.2. In particular, we develop technology to extract, derive and analyze design data from 3D design information in Autodesk Revit Architecture. We will give an introduction to
2.2. 3D CAD model and Autodesk Revit Architecture

This work is collaborated by a multi-disciplinary team including two professors and their students: Dr. Sheryl Staub-French from civil engineering who provides domain knowledge, and Dr. Rachel Pottinger from computer science who leads the evaluation and implementation of data integration.

2.2 3D CAD model and Autodesk Revit Architecture

Computer-Aided Design (CAD) systems are used as advanced design, drafting and drawing management tools by designers. A 3D CAD model is a three-dimensional representation of the design objects created by designers with the help of computers. The use of 3D CAD models is gaining popularity to design buildings. Architects and engineers create one or many 3D CAD models of a building using specialized softwares. Autodesk Revit Architecture ([3]), a software produced by Autodesk company, is one of them that has been widely used for building information modeling (BIM), and we chose it to study for our research.

The version we are using is Revit MEP 2008, where MEP stands for mechanical, electrical, and plumbing. We will use Revit short for Autodesk Revit Architecture or Revit MEP 2008 in the rest of the thesis. Figure 2.3 gives a snapshot of Revit MEP 2008 with a 3D view of a building model.

The native Revit file format is RVT. However, this is not a machine-readable format that could be used for analysis; there is no library provided to access it. However, Revit does have several export mechanisms through which most of the design data can be obtained from certain formats of exported files. More detailed discussion about it will be given in Chapter 3.

2.3 XML and XML standards

XML and XML standards are the main concepts discussed in this thesis. Before going to the main discussion part, we will first give some basic introductions in this section.

2.3.1 XML

XML (eXtensible Markup Language) is a World Wide Web Consortium (W3C) recommended language widely used for storing and exchanging data. It was initially designed and had been used for information exchanging over
2.3. XML and XML standards

the web. Currently, however, XML is generally used to exchange data between all the applications, as a result of its well-known extensibility and flexibility.

An XML file contains a set of elements which are represented by tags. Each element can have a set of sub-elements indicating the relationships among them. An element, including sub-element, can also have attributes with their values to describe simple properties.

The left part of Figure 2.4 provides a piece of XML example where “XMLExample” is the root node. “Example” and “Content” are two sub-elements of “XMLExample” (let’s call them second-level elements) both of whom have their own sub-elements (third-level elements). Both second-level elements have an attribute named “id” with a unique value “1” (for “Example”) or “2” (for “Content”). An XML file could also be viewed as a tree structure. The right part of Figure 2.4 gives the corresponding document tree of the above example.

It is worthwhile to notice that in Figure 2.4, element “Example” has
2.3. XML and XML standards

Figure 2.4: XML example with its tree view

a third-level element “Content” who has an attribute “ref”. The value of “ref” is the “id” value defined by the other second level element “Content”, showing that there is a reference relationship between these two second-level elements. This is called IdRefs in XML. In this way, this XML file represents not only properties of an XML example but also the relationships among them. A lot of analysis we will introduce in the next chapter are based on IdRefs.

Since all elements in a XML file are user defined, it gives users a large flexibility to define their own structures and tags which carries their own semantics. Therefore, a lot of standards based on XML have been created in various areas used for exchanging data between applications.

2.3.2 XML standards

In this thesis, we define “XML standards” as all XML-based, published, well-designed specifications. It could either be used for information exchange over the web, or between applications. As the world of XML rapidly growing, more and more XML standards have been proposed, for usage in many different industries. Developers and companies are making great efforts on improving existing standards and proposing new standards. In the following chapters, we will give an evaluation on the current status of XML standards through reviews on various industries.
Chapter 3

A case study – Developing an application for civil engineering with XML standards

In this chapter, we will evaluate how XML standards work with civil engineering applications by giving a case study. We will introduce the entire developing process for a civil engineering application, including how to choose a standard to adopt and how it works with the application. We will start from introducing the goals and motivations of the application (section 3.1), the predefined ontology that our application will support (section 3.2) and how we chose a standard based on it (section 3.3). We will then evaluate how the selected standard works (section 3.4) and give a description on the implemented application (section 3.5). We end this chapter by summarizing how XML standards work in civil engineering.

3.1 Goals and motivations of the application

This application is developed for the ARTIFACT project, dealing with design feature analysis. It is used to help designers to analyze their model and extract useful information.

As we introduced in section 2.2, the designers (who are often architects) of a building create their design model (3D CAD model) in Revit. Revit provides several kinds of views of a model including views on the entire 3D model (as shown in Figure 2.3), or views of a certain floor plan (Figure 3.1). Designers can also highlight one component in the model to check all its properties and values (both assigned and unassigned). For example, Figure 3.1 gives the view for the first floor of the 3D model Figure 2.3 shows, as well as a highlighted wall with its property table provided in Revit.

Those functions allow designers do simple analysis for a certain compo-
3.1. Goals and motivations of the application

Figure 3.1: Floor and properties views provided in Revit

In Revit, building designers and estimators always want to learn more about complex model features, such as "how many walls are fire rated in this building", or "which walls have openings that are not filled". In Revit, they can only get the answers by checking each wall manually to count the number of fire rated walls or if a wall has unfilled openings. Thus, a system which can answer complex queries on design features would be very helpful for designers to analyze their model and therefore to make better decisions. The application we developed is a prototype of such a system to support user queries on design models.

As we mentioned in section 2.2, the native format RVT is not a machine-readable format that can be used to query in the application. There are several export mechanisms provided in Revit which export RVT into different
formats with various standards. We have to choose one of them to adopt in our application.

Therefore, the first part of this case study is introducing the process of choosing a suitable standard for our application.

We need to define comparing criteria to choose standards. We have a predefined feature ontology which provides a set of features users might be interested in a design model. Our application’s goal is to extract those features in the ontology from a design model and support querying based on it. Hence, our main criterion is how the standard support the features defined in the ontology. We will first introduce the feature ontology in the following section.

3.2 The feature ontology

The feature ontology is defined by Madhav P.Nepal, who is a Ph.D student of Sheryl Staub-French. Civil engineers are using the ontology “to organize knowledge about construction-specific design conditions in a structured vocabulary, facilitating its exchange and reuse. ([22])” From our perspective of developing the application, the ontology provides the domain knowledge defining which kinds of information are interested in the 3D model that would need to be extracted. Since the ontology is still under development, revisions had been made and more features had been added after we finished our analysis for the application. Here, therefore, we will just introduce a snapshot of the ontology as it existed at the time this thesis was written (Figure 3.1).

The feature ontology describes a set of feature classes, feature properties and relationships among feature classes.

A feature class represents a class of object such as wall or an abstract concept such as an intersection ([22]). As Table 3.1 shows, the ontology defines six feature classes: component, opening, intersection, penetration, design uniformity and alignment. Some of them are concrete classes which could be instantiated; others are abstract classes who have several subclasses. Each feature class has a set of properties that describes its characteristics. A feature class sometimes relates to another feature class by relationship properties. Note that a subclass always inherits all properties and relationships from its superclass.

We now describe some of the interesting features of the ontology:

Definition 1
3.2. The feature ontology

Component: an abstract class represents building components. It has a few subclasses including walls, columns, beams, slabs, etc.

Opening: a class of opening objects located in components. Some of them are filled with other components such as windows and doors while others are empty.

Intersection: represents an abstract concept which occurs when two components connect or intersect with each other. There are several kinds of intersection, such as wall to wall intersection, wall to column intersection, column to beam intersection defined by the intersecting components. An instance of "intersection" relates to two components, both of whom have a property named "intersects with" referring to each other showing their relationships.

Penetration: represents a design condition when a building services element, such as a duct, goes through a building component, such as a wall.

Design Uniformity: a class represents feature consistency on a set of components. The uniformity can be characterized on several features, such as spacing, shape, size, location, etc. Take column spacing, which could be assessed along either the X-axis or the Y-axis, for example. "We say that a group of columns is uniformly spaced along the X-axis if, a) there are at least 3 columns in the group, and b) there is a value delta such that every column in the group is aligned along X direction with at least one other column in the group, and there is a distance of delta between their centers."[22]

Alignment: another feature class which applies on a group of components. It represents how the relative components locate and/or orient with respect to some reference. "We say that a group of columns is aligned along the Y-direction if their centers have the same value for x, and are aligned along the X-direction if their centers have the same value for y."[22]

Table 3.1: Portion of the feature ontology. Table taking from [22]

<table>
<thead>
<tr>
<th>Feature Class</th>
<th>Super Class</th>
<th>Class Properties/Relationships</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature</td>
<td></td>
<td>Name</td>
<td>Abstract super class</td>
</tr>
</tbody>
</table>

Continued on next page
### 3.2. The feature ontology

<table>
<thead>
<tr>
<th>Feature Class</th>
<th>Super Class</th>
<th>Class Properties/Relationships</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>Feature</td>
<td>Has opening; Has penetration</td>
<td>If a component feature instance has the change in property value (e.g., a wall having different heights or thicknesses), identifying each of the values can be relevant to practitioners</td>
</tr>
<tr>
<td>Component</td>
<td>Feature</td>
<td>Contained in the storey</td>
<td>Containment property; the value points to the building storey instance</td>
</tr>
<tr>
<td>Component</td>
<td>Feature</td>
<td>Intersects with</td>
<td>Association relationship that links a component in question (source) with other component (target)</td>
</tr>
<tr>
<td>Component</td>
<td>Feature</td>
<td>Includes component parts</td>
<td>Decomposition (whole to part relationship)</td>
</tr>
<tr>
<td>Wall</td>
<td>Component</td>
<td>Is external; Is interior;</td>
<td>Applies to walls whose value of &quot;Is clipped&quot; is true</td>
</tr>
<tr>
<td>Wall</td>
<td>Component</td>
<td>Is load bearing; Is curved; Is clipped</td>
<td></td>
</tr>
<tr>
<td>Wall</td>
<td>Component</td>
<td>Height; Length; Thickness; Area</td>
<td></td>
</tr>
<tr>
<td>Wall</td>
<td>Component</td>
<td>Fire-rating</td>
<td></td>
</tr>
<tr>
<td>Wall</td>
<td>Component</td>
<td>No. of clippings</td>
<td></td>
</tr>
<tr>
<td>Wall</td>
<td>Component</td>
<td>Wall category</td>
<td>Designer specified wall type</td>
</tr>
<tr>
<td>Wall</td>
<td>Component</td>
<td>Curvature</td>
<td>Applies to walls whose value of &quot;Is curved&quot; is true</td>
</tr>
<tr>
<td>Column</td>
<td>Component</td>
<td>Height; Depth; Width</td>
<td></td>
</tr>
<tr>
<td>Column</td>
<td>Component</td>
<td>Size</td>
<td>(Width, Depth)</td>
</tr>
<tr>
<td>Column</td>
<td>Component</td>
<td>Shape</td>
<td>e.g., Rectangular, Square, Round</td>
</tr>
<tr>
<td>Beam</td>
<td>Component</td>
<td>Depth; Length; Width</td>
<td></td>
</tr>
</tbody>
</table>

Continued on next page
### 3.2. The feature ontology

Table 3.1 – continued from previous page

<table>
<thead>
<tr>
<th>Feature Class</th>
<th>Super Class</th>
<th>Class Properties/Relationships</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slab</td>
<td>Component</td>
<td>Thickness</td>
<td></td>
</tr>
<tr>
<td>Slab</td>
<td>Component</td>
<td>Area</td>
<td></td>
</tr>
<tr>
<td>Slab</td>
<td>Component</td>
<td>Bottom elevation</td>
<td></td>
</tr>
<tr>
<td>Slab</td>
<td>Component</td>
<td>Shape</td>
<td>e.g., Triangular, Rectangular, Polygonal</td>
</tr>
<tr>
<td>Opening</td>
<td>Feature</td>
<td>Opening type</td>
<td>e.g., door, window, recess etc.</td>
</tr>
<tr>
<td>Opening</td>
<td>Feature</td>
<td>Length; Height/Width</td>
<td>Height/Width depends on the component where an opening exists</td>
</tr>
<tr>
<td>Opening</td>
<td>Feature</td>
<td>Size</td>
<td>(Length; Height/Width)</td>
</tr>
<tr>
<td>Opening</td>
<td>Feature</td>
<td>Vertical Location</td>
<td>ditto</td>
</tr>
<tr>
<td>Opening</td>
<td>Feature</td>
<td>Located on the component</td>
<td>Association relationship between the opening and the component that hosts it. The value points to component instances</td>
</tr>
<tr>
<td>Intersection</td>
<td>Feature</td>
<td>Angle of intersection</td>
<td></td>
</tr>
<tr>
<td>Wall intersection</td>
<td>Intersection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wall-Column</td>
<td>Intersection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Beam-Column</td>
<td>Intersection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wall-Slab</td>
<td>Intersection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Column-Slab</td>
<td>Intersection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Beam-Slab</td>
<td>Intersection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wall-Beam</td>
<td>Intersection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Penetration</td>
<td>Feature</td>
<td>Type of penetration</td>
<td>Duct, pipe etc.</td>
</tr>
<tr>
<td>Penetration</td>
<td>Feature</td>
<td>Length; Width</td>
<td>(Length, Width)</td>
</tr>
<tr>
<td>Penetration</td>
<td>Feature</td>
<td>Size</td>
<td>ditto</td>
</tr>
<tr>
<td>Penetration</td>
<td>Feature</td>
<td>Vertical location</td>
<td></td>
</tr>
<tr>
<td>Penetration</td>
<td>Feature</td>
<td>Located on the component</td>
<td>Association relationship between the penetration class and the component that hosts it. And, the value points to component instances</td>
</tr>
</tbody>
</table>

Continued on next page
### 3.3. Candidate standards

As we introduced before, Revit provides several exporting mechanisms, which can export native RVT files into a few other formats, including DWG, image, IFC, Relational Database, DWF, and gbXML. Some of them, such as image, are not machine-readable standards; some of others, such as DWG, have read/write libraries which are not free. Under those restrictions, we selected four from all exported formats, as our candidate standards for our application. One of them is a relational database (Microsoft Access), while the other three are (gbXML), or can be treated as (IFC and DWF) XML based standards. We will give an introduction to each of them as well as an evaluation on how they support the feature ontology in the following subsections.

#### 3.3.1 Relational database (Microsoft Access)

Revit provides an option that exports a RVT file into a Microsoft Access database using ODBC. The exported Access file contains a set of tables for

<table>
<thead>
<tr>
<th>Feature Class</th>
<th>Super Class</th>
<th>Class Properties/Relationships</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design uniformity</td>
<td>Feature</td>
<td>Type</td>
<td>Horizontal or vertical uniformity</td>
</tr>
<tr>
<td>Of spacing</td>
<td>Design uniformity</td>
<td>Type of spacing</td>
<td>c/c or clear spacing</td>
</tr>
<tr>
<td>Of column spacing</td>
<td>Of spacing</td>
<td>Direction</td>
<td>X, Y, or both X &amp; Y direction</td>
</tr>
<tr>
<td>Of shape</td>
<td>Design uniformity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>In columns' shape</td>
<td>Of shape</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Of size</td>
<td>Design uniformity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>In openings' size</td>
<td>Of size</td>
<td></td>
<td></td>
</tr>
<tr>
<td>In columns' size</td>
<td>Of size</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Of location</td>
<td>Design uniformity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>In openings' location</td>
<td>Of location</td>
<td></td>
<td></td>
</tr>
<tr>
<td>In walls' location</td>
<td>Of location</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alignment</td>
<td>Feature</td>
<td>Type</td>
<td>Horizontal or vertical alignment</td>
</tr>
<tr>
<td>Column alignment</td>
<td>Alignment</td>
<td>Direction</td>
<td>X, Y, or both X &amp; Y direction</td>
</tr>
</tbody>
</table>
3.3. Candidate standards

all basic component categories in Revit (e.g., walls, doors, floors, etc) as well as tables for types of those basic component categories. For example, the table “Wall” is a table for basic component category “wall” whose columns are a set of properties of this category, including “Id”, “Typeld”, “Volume”, “Area”, “Comments”, “Length”, “TopOffset”, “BaseOffset”, “BaseConstraint”, “UnconnectedHeight”, and so on. Rows in this table are instances of “wall” category in the original Revit model with corresponding values for each property, if applicable. There is another table named “WallTypes” whose rows are types of the “wall” category, which is independent of any specific models. The columns of “WallType” are further properties of each type, such as “Id”, “FamilyName”, “TypeName”, “FireRating”, “Cost”, “Width”, etc. Note that the column “Typeld” of table “Wall” is a foreign key of table “WallType” that refers to further type information of a wall instance.

Figure 3.2: Access sample data from model “Single_Room_With_Beam”

Figure 3.2 is a part of Access file which is exported from a model “Single_Room_With_Beam”. Part (a) is a portion of table “Wall” with all its six rows each of which represents a wall instance in the model. Part (b) is a snapshot for the first 6 rows of table “WallType”. It is showed, for example, the wall whose “Id” is “137214” in table “Wall”(a), has a wall type “250” in table “WallType”(b). Therefore, values of some properties of the wall instance, such as area, length (a) and type name, width (b) can be queried from these two tables.

By reviewing the schema of the exported Access database, we found
3.3. Candidate standards

out that the exported file contained all components in the feature ontology, and a large part of their properties, such as dimension properties, and type information. Moreover, information about “Opening” is also completely provided. However, a few shape properties of wall component cannot be found, such as “Is curve”, “Is clipped” as well as “Number of clippings” and “Curvature”. In addition, most feature classes and properties related to relationships among components are missing, except “Opening” and “Design uniformity of shape or size”. For example, there is not a property related to “Intersect with” for a wall instance. That means features about intersection cannot be explicitly queried from the database. Furthermore, neither absolute nor relative locations of components are provided, results in no deriving way for intersection. There are properties of vertical locations for each component provided, such as “TopOffset” and “BaseOffset”, but without horizontal ones, which are still not enough to derive relative position between components. Similar problems exist for feature class “Penetration” and “Design uniformity of spacing or location”.

In sum, when exporting the original design model into Access database, almost half of all data needed in the feature ontology is missed, leading to the capability of supporting the ontology very limited. We will provide a comparison chart (Table 3.2) in section 3.3.3 showing exactly what is supported and what is missed in the feature ontology for each standard.

3.3.2 XML based standards

gbXML
gbXML (Green Building XML) is an open XML schema developed to facilitate the transfer and integration of design models in building industry ([12]). It is the only XML standard to which a RVT file can be exported from Revit directly.

Since one of the main purposes of gbXML is to support design analysis such as building energy analysis, the exported gbXML file contains only a subset of information important to construction practitioners. The building model in gbXML is represented by a set of space elements and a set of surface elements. Thus, only components related to spaces and surfaces are included. Of all the components in the feature ontology, only walls, windows and doors are exported, while columns, beams, slabs, ducts are lost, leading to feature classes such as intersection, penetration, most types of design uniformity and alignment are not able to derive. In terms of exported components, there are also a lot of properties defined in the feature
3.3. Candidate standards

Figure 3.3: Portion of tree view of exported gbXML file from model "Single_Room_With_Beam"

ontology that are missed in the gbXML file. Take the component “Wall” for instance; only length, width, height, is external, has opening and wall location are exported, while others are lost. In terms of opening, an opening instance is a subelement of the wall it is located on, indicating the opening relationship between them. Since there are no columns, beams and slabs in the gbXML file, only openings located on walls are supported.

Figure 3.3 gives the tree view of a single element of the exported gbXML file from the same model as Figure 3.2. This element represents a wall instance in the model. The right part gives the corresponding values of each attribute or element. It is shown that properties such as height, width, is external and has an opening “NonSlidingDoor” are explicitly provided as a
3.3. Candidate standards

subelement or an attribute, and the location of the wall could be implicitly derived by a set of cartesian points that constitute all vertex of the wall's base plane.

In sum, the exported file based on gbXML standard only remains a very small part of information in the original model; most features in the ontology are missed or not able to be derived.

DWF

DWF (Design Web Format) is a format that developed by Autodesk for distributing design data. Revit provides an option that exports RVT model into DWF file. The design in a DWF file can be viewed using Autodesk's free DWF viewer.

DWF files can also be treated as a ZIP file format ([2]). When we renamed a DWF file into a .zip file and unzip it, we find that the package contains a set of DWF metadata files (in XML) and resource files (in various formats) ([2]). Among those files, there is one with extension “.content.xml” which is the XML file describing content information in this DWF packages exported from the design model. We treat this “DWF-content” XML as one of the candidate standard for our application.

The schema of DWF-content XML is not complex. Every basic building category, family type, sub type and instance in each sub type that contained in the original model is an element named of “dwf:Object”. The relationships among these four kinds of “dwf:Object” elements are representing by a tree structure. Figure 3.4 shows the tree view for a “dwf:Object” of basic component “Wall” and all wall instances belonging to that category. All building components, including walls, columns, beams, and slabs in the model are defined with their categories in this way.

Most properties of components are clearly linked to instances as sub elements “dwf:Property” of “dwf:Object”, as Figure 3.4 also shows. DWF-content contains a large part of properties of component defined in the feature ontology, such as length, width, type, is external, etc. However, similar to Access, it also misses properties about complex shape (curved and clipping) as well as most features related to locations, including intersections, penetrations, and uniformity in location. Those features are neither explicitly provided, nor able to be derived as a result of missing location information. Figure 3.5 gives a piece of DWF-content XML data for a wall instance. It shows what properties related to wall components are provided.

To sum up, the capability of DWF-content XML to support the feature ontology is very similar to that of Access. Most basic components and
3.3. Candidate standards

Figure 3.4: Tree view of “Wall” category in DWF-content XML

their properties are provided but a lot of features related to component relationships are lost.

ifcXML

IFC (Industry Foundation Classes) is an object oriented file format with a data model developed by the International Alliance widely used for modeling in building industry. ifcXML is its corresponding XML specification. Both IFC and ifcXML specifications are open and officially published ([11]). There is an option in Revit that exports a RVT file into IFC directly. We then use an IFC/ifcXML specifications (IFC 2x2) based convertor to convert the
3.3. Candidate standards

Compared to the other three standards, the exported file based on ifcXML contains the most information defined in the feature ontology. Besides all components and simple properties such as dimensions and types, many complex features and properties could also be derived. For example, the location of a component could be decided based on information about a certain point, orientation and dimensions. As long as the locations of relative components are given, relationships such as intersection and penetration could also be derived. Moreover, ifcXML also contains complex shape information (curved or clipped) which are not supported by exported files in any other standard.

However, there are still challenges with ifcXML. Among them are: (1) information about MEP components such as ducts is not complete, so that feature class “penetration” cannot be supported; (2) not the orientations of all components are provided. Some special ones such as the directions of curved walls are missed, result in only a part of relationships that are based on locations are available; (3) compared to the other three, ifcXML are in the largest size and most complex schema. Properties are often not attached.
3.3. Candidate standards

to a component directly, but related indirectly through IdRefs. Some of the reference paths are very long.

Figure 3.6: Hierarchical representation of exported ifcXML file

Figure 3.6 gives portion of hierarchical representation of the exported ifcXML file. The right part of the figure shows that it needs to relate five elements through IdRefs to identify a single opening relationship between a door and a wall, while four to attach simple properties such as dimension or “is external” to a wall.

Although ifcXML is much more complex, to summarize, it provides the most information which gives it the largest capability to support the feature ontology.
3.3. Candidate standards

3.3.3 Comparisons

In this section, we will summarize all the four standards based on their capability supporting the feature ontology. We give a comparison chart 3.2 showing exactly what are supported and what are not supported for each standard.

In this table, “Y” denotes to “supported”; “N” denotes to “not supported”; and “P” denotes to “partially supported”, meaning that the property is sometimes but not always supported by the standard. For those properties supported by a standard, part of them can be explicitly queried from the exported file, while a large other part of them can only be implicitly derived by querying, calculating or combining some other explicit ones. The complexity of deriving those properties is various.

It is worthwhile to mention that there are some of the features, such as sizes for unfilled openings, penetrations and slab intersections, are not supported by any standard we have reviewed. In addition, there are also a couple of features, such as unfilled opening instances and intersections (except ones with slab), which are only able to be derived by ifcXML rather than any other standards.

Table 3.2: A comparison on how the four selected standards – Access, gbXML, DWF (stands for DWF-content XML) and ifcXML – support the feature ontology

<table>
<thead>
<tr>
<th>Feature Class</th>
<th>Class Properties/Relationships</th>
<th>Access</th>
<th>gbXML</th>
<th>DWF</th>
<th>ifcXML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>Has opening</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>Y</td>
</tr>
<tr>
<td>Component</td>
<td>Has penetration</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Component</td>
<td>Contained in the storey</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Component</td>
<td>Intersects with</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>P</td>
</tr>
<tr>
<td>Component</td>
<td>Includes component parts</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>P</td>
</tr>
<tr>
<td>Wall</td>
<td>Is external</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Wall</td>
<td>Is interior</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Wall</td>
<td>Is loadbearing</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Wall</td>
<td>Is curved</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Wall</td>
<td>Is clipped</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Wall</td>
<td>Height</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Wall</td>
<td>Length</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Wall</td>
<td>Thickness</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

Continued on next page
### 3.3. Candidate standards

<table>
<thead>
<tr>
<th>Feature Class</th>
<th>Class Properties/Relationships</th>
<th>Access</th>
<th>gbXML</th>
<th>DWF</th>
<th>ifcXML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wall</td>
<td>Area</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Wall</td>
<td>Fire-rating</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Wall</td>
<td>No. of clippings</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Wall</td>
<td>Wall category</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Wall</td>
<td>Curvature</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Column</td>
<td>Height</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>P</td>
</tr>
<tr>
<td>Column</td>
<td>Depth</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Column</td>
<td>Width</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Column</td>
<td>Size</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Column</td>
<td>Shape</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Beam</td>
<td>Depth</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Beam</td>
<td>Length</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Beam</td>
<td>Width</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Slab</td>
<td>Thickness</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Slab</td>
<td>Area</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Slab</td>
<td>Bottom elevation</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Slab</td>
<td>Shape</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Opening</td>
<td>Opening type</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>Y</td>
</tr>
<tr>
<td>Opening</td>
<td>Length</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Opening</td>
<td>Height/Width</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Opening</td>
<td>Size</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Opening</td>
<td>Vertical Location</td>
<td>P</td>
<td>N</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Opening</td>
<td>Located on the component</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>Y</td>
</tr>
<tr>
<td>Intersection</td>
<td>Angle of intersection</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Wall intersection</td>
<td></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Wall-Column intersection</td>
<td></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>P</td>
</tr>
<tr>
<td>Beam-Column intersection</td>
<td></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>P</td>
</tr>
<tr>
<td>Wall-Slab intersection</td>
<td></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Column-Slab intersection</td>
<td></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Beam-Slab intersection</td>
<td></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Wall-Beam intersection</td>
<td></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>P</td>
</tr>
</tbody>
</table>

Continued on next page
3.3. Candidate standards

Table 3.2 – continued from previous page

<table>
<thead>
<tr>
<th>Feature Class</th>
<th>Class Properties/Relationships</th>
<th>Access</th>
<th>gbXML</th>
<th>DWF</th>
<th>ifcXML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Penetration</td>
<td>Type of penetration</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Penetration</td>
<td>Length</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Penetration</td>
<td>Width</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Penetration</td>
<td>Size</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Penetration</td>
<td>Vertical location</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Penetration</td>
<td>Located on the component</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Uniformity of column spacing</td>
<td></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Uniformity in column’s shape</td>
<td></td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Uniformity in opening’s size</td>
<td></td>
<td>Y</td>
<td>N</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Uniformity of column’s size</td>
<td></td>
<td>N</td>
<td>P</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Uniformity of opening’s location</td>
<td></td>
<td>N</td>
<td>P</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Uniformity of wall’s location</td>
<td></td>
<td>N</td>
<td>P</td>
<td>N</td>
<td>P</td>
</tr>
<tr>
<td>Column alignment</td>
<td></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>

3.3.4 Our decision—choose ifcXML

Table 3.2 shows that the exported file based on ifcXML standards are potentially able to support the most features in the feature ontology. None of the other three can support more than half of the features or properties defined in the ontology. Although the schema of ifcXML is the most complex one of them, and deriving features and properties is also more complicated than any of the others, we still decided to choose ifcXML as the standard we would adopt in our application, since our main goal is to extract the most features defined in the ontology from the original model.

In the next section, we introduce how to apply ifcXML in our application and how it works.
3.4 Using ifcXML for the application

3.4.1 How to extract ifcXML files from Revit

As introduced in the previous section, we export the 3D model to an IFC file directly using the export mechanism provided by Revit. Then we convert the IFC file into an ifcXML file, using a convertor developed by San Chang, who is a previous student of Computer Science department, UBC. We will call this convertor “San’s convertor” in the rest of this thesis. Revit supports exporting 3D models into two IFC versions: 2x2 and 2x3. The latest IFC/ifcXML version published by IAI is IFC2x3 (IFC2x Edition 3)/ifcXML2x3. However, San’s convertor is based on version 2x2. Therefore, we chose IFC2x2/ifcXML2x2 specifications to use for our application.

3.4.2 Observations and analysis on ifcXML: complexity, content, and challenges

A solid understanding of the ifcXML schema and its hierarchy is indispensable before we can extract information from ifcXML. This section will give several key observations on ifcXML from our analysis ([23]).

We mentioned in Section 3.3.2 that the schema of ifcXML is very complex and properties are often not attached to a component directly. Figure 3.7 (a) to (c) shows a 3D wall component, a hierarchical representation of some ifcXML elements in an XML viewer, and the actual ifcXML respectively. It shows in (c) that under the element tag “IfcWallStandardCase” which represents a wall instance, only very limited information can be explicitly found through attached attributes or subelements (e.g., “ObjectType”, which may contain external, fire-rating and size of the wall). A large number of properties and relationships are attached to objects implicitly by IdRefs. It becomes apparent that analyzing how objects are linked with different properties and relationships is often the first, complicated necessary step to query features from ifcXML data. This work, which requires expert knowledge on IFC/ifcXML, is time consuming due to the large size and complex schema of ifcXML.

Figure 3.8 shows a few concepts and the IdRef paths showing their linkage with a wall object. It is obviously shown that to attach a property/relationship to an object is not straightforward but often need several ID referring steps. For example, a door opening is attached to a wall by referring related IDs to four related elements (Figure 3.8 (b)). Figure 3.9 gives the actual ifcXML pieces related to “linking a door to a wall” and
3.4. Using ifcXML for the application

Figure 3.7: A 3D wall object with corresponding ifcXML representations

points out the referring paths. It shows that five IFC elements have been involved to identify a single opening-component relationship.

However, even through IdRefs, not all properties and relationships can be extracted from ifcXML. There are a large part of properties and relationships not explicitly defined in ifcXML. We have to derive them through analyzing some other explicitly provided ones. For instance, wall property “Is clipped” is not defined in ifcXML. We derive it through some analysis on wall shape representations. We will explain some details about how to query and derive those properties in Section 3.4.3, by giving several concrete query examples. Table 3.3 lists some properties/relationships supported by ifcXML which are explicitly provided that we can easily query, and those that are not defined
3.4. Using ifcXML for the application

Figure 3.8: Properties and relationships with IdRef paths showing their linkage to a wall object in ifcXML

and we have to figure out ways to derive them.

Challenges

We discussed in section 3.3.2 that there are still a certain part of features/properties defined in our feature ontology not supported by ifcXML currently (see Table 3.2 which lists what are supported and what are not

<table>
<thead>
<tr>
<th>Provided</th>
<th>Properties/Relationships</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contained in the storey, Is external, Is interior, Is loadbearing, Height, Length, Thickness, Area, Fire-rating, Wall category, Intersects with Wall</td>
<td></td>
</tr>
</tbody>
</table>

| Derived | Has opening, Is curved, Is clipped, Curvature, Intersects with Column, Intersects with Beam, Uniformity of wall's location |

Table 3.3: What is provided and what is derived in ifcXML— for Wall component
3.4. Using ifcXML for the application

Figure 3.9: Linking "Door Opening" to a Wall in ifcXML
3.4. Using ifcXML for the application

Here we give several problems existing in ifcXML which cause its limitation.

1. Most information for un-filled openings is missed. There are two kinds of openings: one is filled by another component (e.g., a door or a window), the other is void. All properties of filled openings, including dimensions, “Located on the component”, are provided in ifcXML; while most ones of un-filled openings, such as “length”, “height/width”, “size” and “vertical location”, cannot be found. This causes that most properties of “Opening” are only partially supported in ifcXML.

2. Component parts for some walls are missed. We observed that for those walls composed by several parts, there are part of them whose component parts can be found while for some others their component parts are lost. We have not figured out if there is a particular type of wall whose component parts are not supported by IFCxml.

3. The orientation of curved walls cannot be found. Usually, for each wall, ifcXML provides an insertion point (the insertion point of an object is the point at which the designer started to draw the object in Revit’s graphical interface) and an orientation to determine the location of the wall. The locations of components are crucial to determine relationships among components such as intersections and penetrations (see Section 3.4.3, Query 2 for details). Without the orientations of curved walls, we cannot derive intersections and penetrations related to them.

4. “No. of clipings” for wall component are not supported so far. “Is clipped” is not a predefined property in ifcXML. We derive it through analyzing the shape representations (see Section 3.4.3, Query 1 for details). However, the shape representations is too complex and various that we have not figured out how to derive the number of clipings so far.

5. There are two basic categories for columns: “StructuralColumn” and “Column”. The height for “Column” cannot be derived. There is no “Height” property for both kinds of column defined in ifcXML. We have to derive it mathematically. For “StructuralColumn”, we calculate its height by using formula \( \text{volume}/\text{area-of-bottom face} \). However, for “Column”, no volume information is provided causing this method failed. We also tried a few other ways, none of whose required data
3.4. Using ifcXML for the application

has been completely provided in ifcXML. Due to the lost of the height information for one kind of column, “intersections” related to column, which requires the height of column (see Section 3.4.3, Query 2 for details), can only be partially supported in ifcXML.

6. “Shape” of slabs cannot be determined so far, results in the exact location of a slab cannot be derived. Without the location, we are not able to determine if a slab intersects with other components. Therefore, slab intersections cannot be supported.

7. Information about ducts is really limited. Only type, insertion point and shape (shape is really complicated to get though) are provided. Neither dimension nor relationship information can be found. That results in that all properties (e.g., “length”, “location”) and relationships (e.g., “Has penetration” of a component, “Located on the component” of a penetration) about duct penetration not able to be supported. It is necessary to mention that we have only analyzed duct penetration so far, not included conduit or piping ones.

8. It is not always clear that where the “insertion point” is. We mentioned above that the insertion points is crucial to determine the location of a component. We currently assume that all insertion points have their certain place which is as default. For example, we assume that insertion points of walls are always in the middle of the lower-left edge and insertion points of columns are in the center of the bottom face. However, this is not always true. It is possible that designers change the location of insertion points to where they prefer, when they create their models. If we cannot determine where the insertion points are, all deriving methods based on them would fail.

We notice that most of those challenges are related to spatial data. We define spatial data including all geometric (e.g., dimensions) and topological (e.g., intersection, adjacency) data. Most problems listed above will be solved if we are able to extract all spatial data from IFC/ifcXML. We found out that IFC/ifcXML does contain such data in some manner: geometric data is usually represented explicitly (through Figure 3.8 path (a)), except a small part such as those mentioned in challenges 1, 5, 7; while most topological data is not explicitly represented. We derive topological information basically by analyzing locations of relative objects (see Section 3.4.3, Query 2 for example). Currently, the location of an object is determined by a reference point (insertion point in our case), orientation, dimensions
3.4. Using ifcXML for the application

and shape of the object. We extract all information above from ifcXML to decide the locations of relative objects in order to calculate topological properties. However, sometimes not all required information is available in ifcXML, e.g., exceptions listed in challenges 1 and 3 to 8. That is the main problem causes those “Not Supported” properties in Table 3.2, currently.

To solve this problem, we noticed that ifcXML provides some representation of surface shape for each object. This is potentially another way to derive spatial information from ifcXML. However, this surface shape is represented in a really hard manner that is not easy to be used. Section 3.4.3, Query 1 talk a bit on how to analyze this surface shape through path (g), (h) in Figure 3.8. This is only a small trial we have done while a deeper understanding on its representation and complex spatial geometric methods needed are still under investigation.

IFC Engine Viewer

People did make efforts on extracting IFC information. IFC Engine Viewer ([34]) is a free IFC browser developed by TNO, Germany. Figure 3.10 gives a snapshot of opening a IFC file using IFC Engine Viewer. The left part is a tree view of all components in the model; the right part gives a 3D view and properties of the selected component. We can see that it maps all simple properties to the components and even categorizes them. For example, in terms of a wall, we can see “Pset_WallCommon” properties including Reference, LoadBearing, ExtendToStructure and IsExternal; “PSet.Revit.Constraints” properties including Base Offset, Base is Attached, Top Offset, etc; “PSet.Revit.Dimensions” properties including Length, Area, Volume, etc., and their values.

It also contains nodes such as "Axis" and "Body" under a component instance in the tree, but no value is provided. Therefore, we still cannot browse location and shape information in the viewer. Moreover, relationships such as intersection, penetration are not provided as properties.

This observation verifies our discussion that there are some properties that can be simply extracted from IFC/ifcXML, while other properties, such as locations, are hard to access. It also demonstrates that however, IFC does represent locations in some manner as it supports the 3D representation in the viewer. We have to figure out ways to derive them.
3.4. Using ifcXML for the application

3.4.3 Querying processing

In this section we take some selected query examples to illustrate the query process and the difficulties encountered in the process ([23]).

Query 1: What is the total length of all walls for each wall type and shape?

The ontology in Section 3.2 differentiates walls as two types: interior and exterior, and considers the shape to be either curved or clipped. Therefore, we use the same set of distinctions. To process this query, we first cluster
3.4. Using ifcXML for the application

walls by their type and shape, then derive the length of each wall, and finally calculate the total length for each wall type and shape.

- **Step 1: Identify the Wall Type - Interior vs. Exterior.**
  ifcXML contains a Boolean property named "IsExternal". It is set as "1" and "0" for external and interior walls respectively. However, this property is not explicitly attached to a wall (i.e., it is not as an attribute or sub-element of a wall). We have to query it through IdRefs. In ifcXML, each object, property and property set is treated as an IFC element having an "ifcID". Most simple properties, including "interior or exterior", are linked to an object by attaching their ifcIDs to it, through the path shown in Figure 3.8 (a). Figure 3.11 gives the actual ifcXML related and how they linked together.

- **Step 2: Identify the Wall Shape - Clipped and Curved.**
  We defined clipped and curved as wall properties. They are however, not explicitly defined in ifcXML. Deriving them requires an understanding of how walls are represented in the IFC model. There are two ways of representing walls in ifcXML: "IfcFaceBasedSurfaceModel" and "IfcExtrudedAreaSolid" are shown in Figure 3.8 (g) and (h) respectively. IfcFaceBasedSurfaceModels contain IfcConnectedFaceSets and, following several more referring steps, we eventually find the points that are used to define a wall. The faces, however, are not in consistent shape and order, which makes it too difficult to tell if a wall is clipped or curved. The second representation, as shown in Figure 3.8 (h), defines a shape by sweeping a bounded planar surface. The planar area, as well as the direction and the length of the extrusion are given. The planar area can be a rectangle or it could be composed of lines or curves. Based on our observations thus far, it appears that all non-clipped walls are represented using an IfcExtrudedAreaSolid, and clipped walls are represented using an IfcFaceBasedSurfaceModel which are more complicated to analyze. Therefore, we decide whether a wall is clipped by checking whether the shape is represented using an IfcExtrudedAreaSolid. If that is the case, the wall is not clipped otherwise it is clipped. However, there may be exceptions to this rule.

It is difficult if not impossible to reason about curved walls using the paths shown in Figure 3.8. If the wall is represented using an IfcFaceBasedSurfaceModel, we would need to analyze all faces, which is too complex. On the other hand, if the wall is represented using an
IfcExtrudedAreaSolid, although the planar area can be composed by curves (the path on the right in Figure 3.8 (h)), we would still need to determine if those curves are the longer edges, since only walls whose longer edges are curved are treated as curved walls. In either case using the information listed in Figure 3.8, it is very complicated to derive the property "curved". The alternative to this approach is to look for the shape of the boundary surface of a wall to determine whether or not the wall is curved. Figure 3.8 (d) shows the IdRef path needed to determine the shape of the boundary surface from a wall. If the path ends with an "IfcPolyline", then the wall is not curved; otherwise (i.e., it ends in IfcTrimmedCurve) the wall is curved.
3.4. Using ifcXML for the application

- **Step 3: Find the Length of Each Wall and Calculate the Total.**

  The "Length" property is linked to a wall object similar to the property "IsExternal" described earlier. Replacing the property name "IsExternal" by "Length" provides the length of every wall. The final step is to sum up the lengths of the walls in each cluster to calculate the total length of the walls.

Query 2: Which walls have component intersections (excluding slabs) and penetrations?

- **Step 1: Identify the Intersecting Components - Explicit and Implicit.**

  We consider two kinds of intersections: wall (wall to wall) intersections and wall-column (wall to column) intersections. Wall intersections are typically explicit in ifcXML (because they are modeled explicitly), so we can query ifcXML (following the paths shown in Figure 3.8 (c)) to identify those component intersections. Determining which wall intersections are non-perpendicular, however, cannot be queried directly from ifcXML. They can be derived using the orientations of related walls as shown in Figure 3.8 (f). We first extract these two orientations which are represented by unit vectors, and then use a geometric formula to calculate the angle between them.

  Wall-column intersections are often not explicit in ifcXML because these element connections are not typically modeled explicitly in 3D. Therefore, these types of intersections have to be derived by analyzing the location information of related objects. We use an open source collision detection library called "RAPID" for this purpose. RAPID deduces the connectivity of two objects that are defined as triangular meshes. A triangular mesh comprises a set of triangles that are connected by their common edges which can lead to more efficient operation by graphics software packages and hardware devices. However, it requires some pre-processing since an object in ifcXML is not defined by a triangular mesh. We first extract object attributes including the insertion point, dimensions, and direction of an object as shown in Figure 3.8, and then remodel it into a triangular mesh. Given the vertices of an object - which can be calculated by an object's insertion point, dimensions and directions - we break every rectangle into two triangles to form a triangular mesh. This process makes it more difficult to find intersections between round columns and curved walls. We simulate a curved face by breaking it into several rectangles and then transfer
3.4. Using ifcXML for the application

those rectangles into a triangular mesh. After the pre-processing, we use RAPID to complete the connection detection. Since we cannot get the orientation for curved walls in ifcXML, as previously mentioned, we currently support queries for wall-column intersections involving straight walls. This function can be applied to curved walls if location information becomes available.

• Step 2: Identify the Duct Penetrations.
As previously mentioned (section 3.4.2), duct information is limited in ifcXML. A few properties such as type, insertion point and shape can be found in ifcXML but dimension and relationship data are missing even though they were defined in the 3D model. DWF and relational exports also provide type and dimension data for ducts, but not the location or relationship information. Therefore, we cannot currently support the above query related to duct penetrations. Should the location information of ducts become available; perhaps by using the insertion point and directions similar to other building components, we may follow the same reasoning process used to query component intersections as describe above.

Query 3: Which columns are aligned?
• Option1: Identify columns aligned with a particular column
A construction practitioner may wish to know what columns are aligned with the one being considered in a specific direction (assume the X-direction). This problem can be solved by drawing a horizontal line through the center of the column, and reporting all other columns which intersect with this line. Specific to ifcXML data, in order to answer this query, we need to extract the (x, y) values from ifcXML representing a column’s center. In order to do this, we must follow links as in the previous queries, to obtain (1) the column’s placement attribute (represented by an IfcLocalPlacement element), (2) the specific representation of this placement (an IfcAxis2Placement3D element), and finally (3) the coordinates (represented by an IfcCartesianPoint element). Through querying ifcXML, we can easily find these coordinates and retrieve the answer to our query: all columns having the same y value as the column in question.

• Option2: Identify groups of columns aligned in a specific direction
3.5. The application

In this case, we want to know which groups of columns are aligned in a particular direction (in this case, the X-direction). This is possible by retrieving all columns' y coordinates as done in the previous query, and sort the results by y value. We can then programmatically compute the groups of columns aligned along the X-direction by scanning the resulting list, creating a new group whenever a pair of consecutive non-equal y values is found.

Query 4: Which columns are uniformly spaced?

We cannot simply extract these groups from ifcXML by sorting or grouping on the columns' (x, y) values. To answer this query, we have devised a "sweep line" algorithm. First, we extract the (x, y) values for each column's center (as explained above). If we imagine these values being laid out in a two dimensional space (a plan view of the columns), our algorithm can be visualized as a line which sweeps horizontally (for uniformly spaced columns along the X axis) across this space, computing the uniformly spaced column groups along the way. The algorithm works by maintaining a table of uniformly spaced column groups, indexed by delta (the spacing between columns in the group). Each time the sweep line encounters a column, we look back at the spacing from the previous column aligned in the X-Direction (if one exists), and either update an existing group if the spacing is similar, or make an entry for a new group in the table if the spacing is different. The result is groupings of columns with similar spacing.

3.5 The application

In the previous section, we provided a detailed evaluation on ifcXML and how to use it to support queries based on our ontology. In this section we will give a description on our application and its current version supported by ifcXML.

3.5.1 Overview of querying process

As we introduced in Section 3.1, the application we are developing is aimed to extract features from a 3D CAD model. Since we chose IFC as the standard to use, the goal then becomes deriving features from a IFC model. Madhav P. Nepal proposed an IDEF0 (IEEE standard Functional Modeling Language, [10]) model of the querying process in the system (Figure 3.12,
3.5. The application

There are three major steps in the process, as illustrated in Figure 3.12:

1. Define Project-Independent Features: Users define the features that are important for a particular domain based on the generic feature ontology. The system will provide a generic feature ontology (the one introduced in Section 3.2), on which users can select a subset, or create new feature types and define relevant properties if it is not already available. Those user-defined features will be project-independent which can be reused from project to project.

2. Create Feature Based Model: The system will create a feature-based model (FBM) from the IFC model loaded by the user, as an instance of the user-defined ontology in step 1. This FBM is project-specific represents the design of the model in terms of features that the user care about.

3. Query the IFC-based 3D Model: The system will query the FBM to answer specific questions posed by the user. They will be able to, for example, browse the FBM, do property filtering, or calculation (e.g., what is the total length of curved walls) on the model. This is the
main functionality of the system that allows users to analyze their model through specifying concrete queries.

3.5.2 Limitations and current version

The above section introduces the ideal system that the civil engineers expect. However, the current version being implemented are not be able to support all of them:

- User-defined features (see step 1 above) are not supported so far. As illustrated in Section 3.4, due to the complexity of ifcXML, to extract a feature/property from ifcXML is always not straightforward. The mapping between a feature/property and ifcXML is various and sometimes very complex. It is currently impossible to support user to define any feature/property whose mapping with ifcXML should be automatically generated by the system. Therefore, currently we only provide the fixed, default ontology in the system and all FBM or queries are generated based on it.

- Although using the default ontology, due to those limitations mentioned in Section 3.4.2, we are still not able to support all the features/properties defined in it. We have only implemented those whose deriving ways are already figured out.

- The query language of the system is still undecided. For the query part (step 3 above), therefore, users are not able to create their query in a natural way but through a less flexible customizing method. Section 3.5.4 below will provide some details of the method.

Thus, the current application that will be shown below is only a prototype that gives the basic functions of the system. It is still a long way to develop a final, complete version.

3.5.3 Languages and technologies

Most functions of the application as well as all graphic user interfaces are written in Java. A few other functions, including San's converter and the intersection detection function (See Section 3.4.3, Query 2) are written in C++ and called by main Java programs.

To query XML document, we use the standard XML query language, XQuery ([35]). "XQuery is to XML what SQL is to database tables ([36])."
3.5. The application

Similar to SQL, we embedded XQuery expressions in our JAVA programs. The result of an XQuery expression is the data we need in XML; we then manipulate it and return the final result through Java programs. The following code provides a sample XQuery expression in order to "query the length of an instance whose id is i48" from an ifcXML file. Performing this XQuery expression will return a result shown below, indicating the length we are looking for.

```
"for $doc in doc('test.dbxml/model4'),
  $propSet in $doc/IfcRelDefinesByProperties[RelatedObjects//@ref='i48']
  /RelatingPropertyDefinition,
  $singleProp in $doc/IfcPropertySet[@id=$propSet//@ref]/HasProperties,
  $propValue in $doc/IfcPropertySingleValue[@id=$singleProp//@ref]
  where $propValue/Name='Length'
  return $propValue/IfcPositiveLengthMeasure/text()"
```

Result:

```
30.140654259535
```

However, XQuery is not as mature as SQL. There are still several weaknesses, such as limited expression capability ([7]).

We use Oracle Berkeley DB XML (version 2.3.10), which is "an open source, embeddable XML database with XQuery-based access to document" ([28]) to store our XML files.

Due to the complicated query paths and sometimes requirements of multiple separated queries or functions, extracting features/properties from the original ifcXML is quite time consuming: e.g., determining an intersection requires calculations on properties extracted by several queries and a calling to the external intersection detection function. The performance would not be good if the system queried the original ifcXML for every query the user posted. Therefore, we create an intermediate schema in XML to store information that extracted from ifcXML. Since this intermediate XML is basically represented a FBM (Feature Based Model, defined in Section 3.5.1, Step 2), we call it FBM-xml. The schema of FBM-xml is really simple: every instance of a feature is an element; all properties of a feature with their values are explicitly represented as sub-elements. Therefore, querying this FBM-xml is very straightforward and fast. Figure 3.13 gives a piece of FBM-xml example data showing a Wall instance with all its properties as sub-elements.
3.5. The application

For every IFC model the user loads, the system will first query the ifcXML generated by San's converter, and then create a corresponding FBM-xml automatically. All user queries, including FBM browsing and customized queries, are performed on this FBM-xml. In this way, the long, complex original ifcXML will only be queried once, which improves the performance a lot.

3.5.4 User Interfaces

We will provide a few main user interfaces in the current version of our application.

Figure 3.14 shows the main window of the application. The user can choose to open a new model or load an existing model from the main menu “File”. Then the system will automatically generate a FBM (for a new model) or load the corresponding FBM (for an existing model) in the backend. After the FBM has been generated, the user can browse the FBM in the way Figure 3.14 shows. The left panel in this figure is the tree view of FBM categorized by Level; the right panel shows the properties of the
3.5. The application

instance the user clicked in the tree.

Except browsing FBM, there are two other ways to query the model: customizing a query or performing a specific query from the query library.

Figure 3.15 is the user interface for querying from library. The user can easily choose a predefined query to perform, and get the result.

Figure 3.16 and 3.17 shows the two parts of customized query. The first part is property filtering (Figure 3.16). The user is able to select the feature type, and properties to perform a filtering on instances. For example, Figure 3.16 gives a query “List all wall instances that is external and length less than 40”. When the user submit the query, the result is shown in the left part of Figure 3.17. Sometimes the user may want to do some quantification or grouping on the filtered result. Figure 3.17 gives such options. There is also an option to add this customized query in to library for reuse in the future.
3.5. The application

Figure 3.15: Query from library UI
3.6 Case study conclusion - XML standards for civil engineering

In the previous sections of this chapter, we provided detailed descriptions of a case study. We introduced a whole process of using an XML standard in a civil engineering application. Through this process, we saw that a large part of our hypotheses (Section 1.3) have been verified:

1. **XML has high expressiveness and flexibility:** In our study case, comparing all standards we reviewed, including a relational one and three XML ones, ifcXML - a standard based on XML - contains the most information that we need (Section 3.3). The relational one (Mi-
3.6. Case study conclusion - XML standards for civil engineering

Figure 3.17: Quantification and Grouping UI

crossoft Access) only provides half of all data needed in our application, while ifcXML supports nearly 80% (could be more if we learn more on its schema), thanks to its specific-defined schema to contain information more flexible. For example, a bunch of 3D points are represented as elements in ifcXML, which could possibly be referred to a general point in an object or a specific vertex such as insertion point. Neither of them is able to be provided in Access. This verifies that XML standards have larger capability to carry complex information.

2. The schema of XML standards is sometimes complex: The schema of ifcXML is really complex and not easy to understand (Sec-
3.6. Case study conclusion - XML standards for civil engineering

The schemas of Access, and the other two XML standards we have reviewed (gbXML and DWF-content) are all simpler than ifcXML, but have lower capability to support data we need. ifcXML contains more information, partially because of its more complex schema. As we talked in the example above, ifcXML contains a bunch of 3D points which are not provided in other standards; however, to figure out how those points linked to an object and if they represent an insertion point or a general vertex requires complicated IdRef paths which are derived based on solid understanding on its schema. We have made a lot of effort to study the schema and to figure out how to derive information from ifcXML for our application. We provided a whole section (Section 3.4) to introduce our findings and work on it. However, there are still problems we have not solved due to the very hard manner in which ifcXML represents information, such as how to derive spatial information by analyzing surface shape which are provided in a very complicated way in ifcXML (Section 3.4.2).

3. XML is not efficient enough: The size of ifcXML file is large. The corresponding ifcXML file for a single room is nearly 1MB and for a simple house is already more than 10MB. This factor, along with weakness of the query language (XQuery) and XML database (Berkeley DB XML), lead to a very poor performance of querying ifcXML files and create feature based model in our application. To generate the FBM for a single room model (with only 2 levels, 6 walls, 4 columns, and 11 openings), it takes more than 10 minutes in our test machine; for a simple house model (with only 2 levels, 9 walls, and about 30 openings), it takes more than 2 hours. We have not tested our application in a real building yet, but we can imagine that for a big, multiple-level building, the performance will be even worse.

5. Usually more than one XML standards are available in one area: There are at least three XML standards available in our study case: gbXML, DWF-content xml, and ifcXML (Section 3.3.2). Some of them are not originally in XML, such as DWF-content xml (got from unzipping DWF file) and ifcXML (converted from IFC). There are probably more XML standards available, but we chose these three for reviewing. It is hard to tell the relationship among them: they are neither purely redundant nor complemented, as they are not designed for the same target. Some parts of their content are overlapped while some other parts are complemented.
6. There is no all-purpose XML standard for each area: We have reviewed three XML standards for our application. As we summarized before, neither gbXML nor DWF-content is able to fulfill half of our requirements. Even though we chose ifcXML to use in the end as it supports a larger part of our ontology, we still found out it cannot perfectly support all (Section 3.3.2, Section 3.4.2). We did not find an XML standard which covers all of our requirements for our application. In other words: there is no all-purpose XML standard which is expected to support every application, including our application, for civil engineering.

7. Differences among available XML standards have to be identified: Since there are several standards available, we had to identify the differences among them in order to choose one to apply. Those three XML standards we have reviewed are developed by different organizations with different targets, which results in that their schemas and contents are quite different (see introductions to each standard in Section 3.3.2). We would not be able to decide which one to use unless we understand the advantages as well as weaknesses of each standard, and the differences among them.

8. Identifying differences among XML standards is usually not easy: However, those differences are not explicit that it took us many efforts to study and compare them (Section 3.3.2, Section 3.3.3). We studied the target, schema, and content for each standard. In terms of content, we defined a feature ontology (Section 3.2) as our evaluation framework, according to which we reviewed what each standard provides and what it misses; for those provided, how they are represented (Section 3.3.2). We gave a comparison chart in Section 3.3.3 based on our studies. With those efforts, we were finally able to decide to choose ifcXML to use, which we found out is potentially have the highest capability to support our application.

9. Sometimes none of the available XML standards is suitable: Although ifcXML contains most information and we finally chose it to apply in our application, it does not provides all data we need (Section 3.4.2). Our study shows that none of gbXML, DWF-content xml and ifcXML is able to provide all information required in our application (Section 3.3.2, Section 3.3.3). An integration of several of them, in order to extract information from different sources, is possibly considered in the future. Moreover, due to the efficiency problems mentioned
3.6 Case study conclusion - XML standards for civil engineering

above, we will probably create an own standard to specifically support our application (we already designed one (FBMxml) for intermediate usage, introduced in Section 3.5.3). These will all lead to extra work: take the intermediated FBXML for example, besides schema designing, a transformation mechanism between original ifcXML to FBXML is needed to develop.

There are still a few of our hypotheses have not been verified from this case study:

4. XML standards improve the facility of data exchange among applications: This has not been verified because our case study only relates to one application therefore does not contain data exchange between applications.

10. Coordination on standards is necessary: This has not been verified with the same reason as above: this case does not include study on data exchange between applications.

11. Transformation between XML and relational databases is needed sometimes: This case study does not includes the case that storing data in relational databases and transmitting it by XML (we have not talked about data transmitting). Therefore such transformation is not needed in this case study.
Chapter 4

A review on XML standards for E-business

In this chapter, we will evaluate how XML standards work with e-business applications by reviewing several related papers. We will give a general outline for each selected paper in Section 4.1, and then synthesize and evaluate in Section 4.2 based on those paper reviews.

4.1 Motivations and paper summaries

In the previous chapter, we looked at a case study for civil engineering of how to extract information, particularly using XML. We drew some evaluations on XML working with our application. That lead us to make some hypotheses on how XML works with actual applications. Therefore we explored in some other areas to see if our hypotheses were supported. E-business seems to be the area with the most papers talking about using XML standards. Moreover, e-business is an area where information exchange over the web is so crucial and prevalent that it becomes consequent to study standards for it. Thus, we chose e-business as the second area to explore.

We began by searching for papers that were related to XML standards for e-business. As previously mentioned, there are a large number of papers about XML standards for e-business. We basically have found two kinds of related papers:

1. Many papers were about one specific XML standard - either an introduction, extension or improvement on that standard. [9] is an example that introduces eCo, which is an XML framework for agent-based e-commerce. [6] is another paper in this category that proposes an improvement on the popular e-business standard ebXML. Those papers give detailed introductions and sometimes deep explanations on a specific standard, but fail to provide a wider view throughout the area.

Since our goal is not to go deep to one standard but get general study
4.1. Motivations and paper summaries

for all, this kind of paper is not able to provide sufficient information for our review.

2. The other kind provides findings on studying multiple standards in an area (either e-business or a branch area of e-business). We expected to find some general summaries from those papers which would be helpful to extract comprehensive conclusions. We also expected that descriptions to single standard would be given as well since they are essential for illustrating general evaluations. As they have the potential to provide both specific and general information, we decided to choose papers from this category for reviewing.

We selected five representative papers, which were the most comprehensive ones, from the second kind above. Following we will give a summary for each selected paper, separately.

The first paper we have chosen is Nurmilaakso et al. [25], which gives a general analysis on XML-based e-business frameworks. An "e-business framework" is defined as a "standard for information sharing within and between companies that answers or gives means to answering at least one of the question of what information to share, when and how ([25], [24])", therefore we consider XML-based frameworks as XML standards. The paper basically describes what e-business frameworks are, how to choose frameworks, and gives a conclusion including commonalities, differences and regularities of frameworks. It analyzes 12 e-business frameworks: BPEL, BPML, CIDX, cXML, ebXML, OAGIS, papiNet, PIDX, RosettaNet, UBL, xCBL and XPDL. For comparison, the paper provides a criterion containing three technical key variables (business documents, business processes, and messaging) and four non-technical key variables (industry, drivers, organization, and openness), and set a value for each variable for every selected framework. Two commonalities, several differences and two regularities are drawn after the comparison of frameworks.

Kim et al. [13] is the second paper we have selected. This paper gives a comparison on XML frameworks for B2B e-services. E-services focus on providing services through the Internet, which is "a key part of the value provided by many businesses" ([13]). The papers first introduces what e-service components are, and compares four selected frameworks (eCo, RosettaNet, BizTalk, and E-Speak) in general (including target industries, metadata and ontology, standard XML efforts and legacy support), as well as how those frameworks support the e-service components. For conclusion, the author summaries that "there is no single framework that addresses all elements of the interactions among business partners in a B2B environment ([13])".
4.2. Synthesis - XML standards for E-business

Our next selected paper, Schmitz et al. [31], talks about XML standards for another area in e-business: electronic product catalogs (EPC, e-catalogs). “E-catalogs form the basis for buying decision and the release of order transactions ([31]).” The paper selects 14 standards for analysis: BMEcat, BMEcat preliminary draft, catXML, CIDX, cXML, DATANORM, EAN UCC, eCX, EDIFACT Pricat, EDIFACT Prodat, OAGIS, OCI, RosettaNet, and xCBL. The comparison of standards in this paper is based on three main groups of criteria: the standardization organization, the methodology used in the standardization process, and the content of the standard. In terms of the content, the author proposes a six-level model as the analyzing framework, which contains data types, vocabulary, documents, processes, framework and metamodel (focuses only on the first 4 levels). Several recommendations on how to improve standards to support e-catalogs have been provided after the analysis of selected existing standards.

Schmitz et al. published another paper [30] focusing also on XML standards for e-catalogs, before publishing [31]. This paper, selected as our fourth one, aims at how those XML standards apply different schema languages. It selects five standards (BMEcat, cXML, EAN.UCC, OAGIS, xCBL) and defines seven areas (specification structure, datatypes, XML attributes, elements, inheritance, being unique or key, and miscellaneous) for examination. Four schema languages have been reviewed: DTD, XSD, XDR and SOX. It concludes from the examination that DTD was generally replaced by newer XML schema languages.

The last selected paper Li [17] is the earliest published one among all these five papers. This paper gives a general analysis on XML standards for e-business. It introduces seven standards (BizTalk, CBL, cXML, IOTP, OAGIS, OCF, RETML) with classifying them into five categories. The comparison of standards is based on the complexity of XML DTD from both quantitative and qualitative aspects. It draws in the end that XML standards are not quite mature therefore integration and interoperation among standards is expected.

4.2 Synthesis - XML standards for E-business

In this section we will give a synthesis on our findings in the selected papers, to evaluate how XML standards work for e-business. We list our hypotheses one by one, with a description of how it has, or has not, been supported by those papers.

1. XML has high expressiveness and flexibility: [25] mentions
two types e-business frameworks including EDI-based and XML-based ones. It reveals that among the three technical key variables (business documents, business processes, and messaging), EDI-based frameworks deal with only the business document, while XML-based ones are used in all ([25]). This verifies that XML has the larger capability than EDI. [13] and [17] both state that the property of XML that letting users define their tags therefore define their domain-specific syntax and semantics simplifies the exchange of information. The introduction of XML-based standard Biztalk in [13] and [17] says it "provides a set of tags used in messages sent between applications", which is an example of the expressiveness of XML user-defined tags. [31] also points out that most available e-catalog standards (16 out of 25) are based on XML and, XML Schema (XSD) – the most important schema language of XML – "has become primary due to its high expressiveness ([31])".

2. The schema of XML standards is sometimes complex: [17] provides a table for complexity of DTDs of six selected XML e-business standards. It shows that OAGIS – a standard "dealing with many different types of complex business applications ([17])" – has the highest values for all parameters: 111 for DTD number, 429 for ATTLIST number, 935 for ELEMENT number and 271k for DTD file size ([17]). Those high values reflect the high complexity of the schema of OAGIS. On the other hand, the two simplest standards only deal with one kind of application, e.g., RETML is created just for "exchanging real estate transaction information ([17])"; OCF is used for exchanging catalog data ([17]). Moreover, OAGIS is focused on machine reading that is hard for a human being to read and understand, while RETML and OCF focus on human reading and are "very difficult for a machine to process" ([17]). Those findings support that in order to handle more complex situations and ease machine processing, the schema of an XML standard sometimes is very complex and not easy for human understanding. The complexity of some XML standards is also verified by the statement in [31] that without comprehensive documentation, "the correct understanding of a e-catalog standard’s semantics is difficult".

3. XML is not efficient enough: The large size of XML documents can be proved in [30] which shows that "extensive catalogs with up to hundred thousand products and attached multimedia objects can
4.2. Synthesis - XML standards for E-business

be, not least because of the XML tags, several hundred MB large ([30], [26])”. It also agrees that it is time-consuming to “parsing and importing large XML document [30]”.

4. XML standards improve the facility of data exchange among applications: Almost every paper mentioned this statement. [25] points out that “the purpose of e-business standards is to enable efficient business interactions between business partners”. XML provides such a “common ground” that as long as business application of different business entities can “generate, receive, and process XML document, they are able to communicate and conduct business with one another ([17])”. [13] also says that with user-defined syntax and underlying technologies, XML simplifies the exchange of information. [31] gives evidence with respect to e-catalog that thanks to the “advent of XML as a meta language for defining business vocabularies and machine-readable documents”, the content of product catalogs is able to be “specified and transferred in a standardized way”.

5. Usually more than one XML standards are available in one area: This is obviously verified that all papers we have reviewed list more than one XML standards in the area they focus on: For e-business in general, [25] analyzes 12 standards and [17] introduces 7. In terms of e-catalog, [31] compares 14 different standards while [13] evaluates 5. [31] also points out that there are at least 25 e-catalog standards can be identified and 16 of them are based on XML. [13] is the only paper talking on e-service which lists 4 XML-based frameworks to review. [25] gives a reason for the existence of multiple standards, which is the different requirements between vendors and users: “vendors tend to drive a wider use standard and users a deeper one ([25])”. In terms of the relationship between standards, there is a detailed summary on e-business frameworks provided in [25]. The summary has been drawn through studying the evaluation variables of both technical and non-technical ones mentioned above and their values for each standard. It states that “standards are not compete in all respects but often cooperate with each other”, and competition and cooperation are “not black and white, but there are various shades of grey” ([25]).

6. There is no all-purpose XML standard for each area: This has been supported in all e-business, e-service, and e-catalog:
4.2. Synthesis - XML standards for E-business

- E-business: “Given the fact that electronic commerce is a complex and growing area, no standard can cover all aspects of electronic commerce ([17]).” For example, a customer always sends a purchase order using OAGIS to a manufacturer, while the manufacturer sends the payment business object to a bank using IOTP after it receives and processes the order ([17]). There needs two different standards to handle this single business scenario.

- E-service: “There is no single framework that addresses all elements of the interactions among business partners in a B2B environment” for e-service ([13]). For instance, BizTalk aims to provides “a guidelines for publishing schemas in XML and using XML messages to easily integrate software programs” ([13]); RosettaNet focuses on “standardizing business processes within the IT industry’s supply chain and to facilitate the interactions between partners” ([13]); and eCo promotes “integration of e-commerce services through marketplaces by creating uniform service interfaces” ([13]). They all have different, specific targets that cannot cover all aspects.

- E-catalog: Analysis at document layer of the 6-level evaluation model in [31] shows that “some standard is hardly usable such as cXML, and some are nearly complete models such as BMEcat; but no standard covers all requirements ([31])”.

7. Differences among available XML standards have to be identified: None of those selected papers are focused on developing one application, but giving evaluations for the entire area. Therefore they do not explicitly state the necessity of identifying differences when creating an e-business application. However, this could be implicitly drawn through the above two hypotheses: since there exists multiple XML standards while no one is all-purpose, to understand the differences of them would be indispensable for choosing a right one. As [13] concludes for e-services: “it is important for users to understand these underlying differences as they evaluate the different platforms to create e-services.” This importance could also be supported by the fact that there are a lot of papers, including all the five of our selected ones, which make an effort to give comparisons and evaluations on different XML standards in their areas.

8. Identifying differences among XML standards is usually not easy: All papers we reviewed provide comparisons among standards to
identify their differences. They all have created their own comparison frameworks: [25] identified seven variables and their possible values; [13] gives criteria in both general and e-service component part; [31] developed a 6-level model for evaluation; [30] introduced an examination with seven areas; [17] gives both quantity and quality comparison on schema. To evaluate, values for each criteria have to been identified, which needs a general study or even deep understanding on every compared standard. For example, to identify the “Target Industries & Purpose” in the comparison framework of [13], one needs to review the documentations for every selected standard; in the “document layer” of the 6-level evaluation model in [31], there are 13 requirements that need to check if and to what extent the standard fulfils them. This requires a comprehensive study on the content of each standard.

9. Sometimes none of the available XML standards is suitable: [31] suggested that current e-catalog applications are of a limited area therefore applications able to support complex products and services are expected. In order to extending the capabilities of e-catalog applications, “e-catalog standards need to broaden their product models” ([31]), which indicates that none of the existing standards will appropriately support the extended applications, so that an extension or improvement is necessary. In terms of standards integration, [13] mentions its possibility that “given the open nature of XML standards and the diversity of platforms, it might even be possible that end-users will find it useful to combine platforms to create viable services.” However, as our hypothesis points out, the integration is always difficult, which can be proved by the example given in [17]: taking the same scenario mentioned in 6, if we are going to integrate OAGIS and IOTP, we could find some equivalent or similar definitions such as “ID in IOTP roughly equals QUALIFIER in OAGIS ([17])”; however, there are also non-matched terms such as “CurrCodeType in IOTP does not have an equivalent in OAGIS; DRCR and TYPE in OAGIS do not have equivalence in IOTP ([17])”. These “semantic differences post challenges to integration” ([17]).

10. Coordination on standards is necessary: [31] gives a similar statement for e-catalog that “some coordination between catalog creators and processors is necessary”. Since there are a variety catalog standards for choosing, there have to be agreements “made cover the use of optional data elements, the fixing of enumerations like currencies
or languages, and even the restriction of domains ([31])". Consider the scenario in [17] mentioned in 6 again; it also proves the importance of the coordination: "since a purchase order in XML BOD using OAGIS and a payment business object in IOTP have data related to monetary, it would be ideal if both systems use the same or similar representation for monetary value ([17])." However, neither OAGIS nor IPTP considered this scenario in their design ([17]). Therefore with the different semantics and data structures, if the scenario in the example needs to be supported, there have to be a difficult integration ([17], see illustration in 9 above). This integration is possible for just point-to-point and one application is aware of the standard the other application is using, but really hard with higher scalability ([17]). [30] also points out the complexity of mapping XML catalog data in one format to another that it "needs not only knowledge of the syntax, but likewise of the meaning of the data ([30], [27])", and it is "a problem if the format is documented little or not and an exact specification of the intended semantics is missing" ([30]).

11. Transformation between XML and relational databases is needed sometimes: [30] states that such transformation is "a main task in electronic data interchange between enterprises". A main reason is that "e-business systems connect existing operational information systems, which are based almost exclusively on relational models and database systems ([30])", while in contrast, "the exchange of catalog data is normally based on XML e-business standards ([30])". It also points out that such transformation is possible ([30], [16]), whose quality however, depends on "the meta information that is formalized in the specification of the catalog standard ([30])". This is treated as the motivation of ([30]).

To conclude, in terms of the e-business area, all of our hypotheses have been supported, by at least one reviewed paper as the synthesis above shows.
Chapter 5

More General Discussions

In this chapter, we will give more general discussions by evaluating XML standards in some other areas, including finance, biology and legislation. After giving a brief introduction to each selected paper in Section 5.1.1, we will synthesize our reviews on XML standards for these three areas in Section 5.1.2, followed by a general conclusion about all areas we have reviewed in the end of this chapter (Section 5.2).

5.1 Reviews on other industries

In previous chapters we provided a detailed case study on XML standards for civil engineering (Chapter 3), which resulted in a set of hypotheses; then we verified our hypotheses in the area e-business by comprehensive paper reviews (Chapter 4). Now we would like to explore more areas, to see if we can generalize our hypotheses even further. We chose three areas for reviewing, including finance, biology and legislation, as we found out data exchange is quite an important issue in these industries where having efficient standards is crucial.

5.1.1 Introduction to selected papers

Similar to what we did for e-business, we started our evaluation from reviewing related papers. There are quite a few papers talking about XML standards in these three areas, although not as many as in e-business. We still prefer comprehensive ones which provide general study on multiple standards. However, most of papers we have found, are about one specific XML standard – as the first type of papers we introduced for e-business in Section 4.1. We selected three representative ones for each area, where most of them are comprehensive; the other few are focused on one single standard.

We selected three papers/articles for financial industry: the first paper [5] provides an comprehensive overview of XML in finance: why XML, what were those XML-based standards and what needed to be improved. The second one [19] is an article published in XML.com which gives some general
analysis and also includes introductions to a few available XML standards for financial services. The third one [14] is a commentary which we selected because it provides a list of highlighted points on XML in the financial industry.

For the biological industry, we chose three papers where two of them describe studies on standards within systems biology: one [32] proposes a comparison method for available standards with demonstrating examples; the other [33] gives a general review of standards in this area. The third paper [1] provides an examination of XML when it is used as a data language for bioinformatics, including both advantages and weaknesses.

In terms of legislation, there are few papers providing general reviews therefore two of the selected papers are focused on one single standard: MetaLex for [37] and NormelnRete for [20]. However, we found one paper [18] that gives introductions to a set of available XML standards as well as a comprehensive comparisons among them. This paper is about Workpackage 3 of ESTRELLA project with a goal of defining an “open, general, jurisdiction independent and language independent XML standard to describe legal sources ([18]).”

5.1.2 Synthesis

In this section we will synthesize our reviews on those selected papers to evaluate whether our hypotheses being supported. For each item in the hypotheses, we will go through the three areas one by one.

1. XML has high expressiveness and flexibility:

- Finance: [5] points out that “XML’s flexible tree structure lends itself to the addition of layers of metadata to the underlying data”, and the “addition of contextual metadata” is quite important in finance to make it possible for end users “filtering the information based on well-known topic areas” ([5]). This advantage of XML has also been proved by introducing the flexibility of several XML standards in [5] and [19]: XBRL “allows companies to add their own important items by extending their local XBRL taxonomy ([5])”; FpML allows financial firms extend its schema “by adding on instruments in new datatype libraries ([19])”; moving FIX to its XML version FIXML makes it able to “be carried in any messaging protocol in the message body” which “extends its flexibility and interoperability potential ([19])”. [19] also states the reason that “FpML has many advantages over FIXML” is
5.1. Reviews on other industries

FpML “was developed with the specific aim of using XML” while FIXML is just “an XML representation of FIX that doesn’t take full advantage of the capabilities of XML” ([19]).

- Biology: There is an example provided in [1] about the rich capabilities of XML for linking data: “cross-references between databases could be replaced by a so-called ENTITY with a direct access to the information under a structured format” and “this would avoid the long process of cross-indexing the database entries at each new release” ([1]). [1] also agrees that “XML is highly flexible; adding new elements or attributes may not require modification of the XML data”. Moreover, [33] points out that XML is the most often used representation formalism “to provide a structure in which different kinds of data can be efficiently represented and exchanged between researchers”.

- Legislation: [20] indicates that the reason why they chose to use XML is its expressiveness: with XML it is possible to specify “not quite the rendering characteristics of a document (such as its font name, size, and style, or its margins and alignments), but rather the structures typical of the class to which the document pertains (such as titles, sections, subsections, paragraphs, and so on). In other words, it is possible to provide descriptions and enforce constraints relevant to any arbitrary class of documents, by listing the constituent elements and the structural rules” ([20]).

2. The schema of XML standards is sometimes complex:

- Finance: [5] indicates the complexity of XML by pointing out that with immature XML tools, which “focus too much on the technology and not enough on achieving what users need to achieve as simply as possible”, users have to have solid understanding of XML, including “how it works and how to make it work best”. However, it should be enough for business analysts in a company to be domain specialists, without having to “be trained up at IT experts in XML” ([5]).

- Biology: It is mentioned in [33] that “the complexity of the XML tree structure can vary a lot between formats”. For example, “BSML, INSDseq, and EMBLxml have less than 7 levels while Seqentry uses 26 levels for representing the same information ([33])”, which is obviously very complex and “it makes manual processing and use of general purpose tools more difficult ([33])”.

63
5.1. Reviews on other industries

- Legislation: Most of the standards introduced in [18] have more than one XML-Schemas/DTDs. For example, MetaLex has a main XML-Schema combined of two partial schemas, along with “a separate schema for decisions and for OWL representations”; LexDania has “one meta-schema along with 41 sub-derived XML-Schemas” ([18]). Although [18] does not provide how complex each of those schemas is, it can still be shown in [4] (a paper proposes NormelnRate) that of the three DTDs defined for NormelnRate, two contain about 180 elements while the third one contains 100, with a total number nearly 500.

3. XML is not efficient enough:

- Finance: [5] highly agrees that the size of XML “can make it infeasible to deliver large amounts of financial information (and deliver it in a timely fashion) over the many lower bandwidth connections which exist even in industrialized countries”. Although without XML, bandwidth issues would “still have existed because the number of trades reported daily from the world’s markets is growing exponentially”, “the desire of end users to receive XML is compounding the situation” ([5]).

- Biology: [1] points out the scalability problem of XML that “if XML data are stored in flat files, queries on files will not scale because XML in itself does not provide scalable facilities such as indexing or data clustering ([1])”. That means “parsing should be done on the fly which leads to poor performances ([1])”.

- Legislation: we have not found evidence for this hypothesis in these three papers since none them mentions the performance when exchange legal documents in XML formats.

4. XML standards improve the facility of data exchange among applications:

- Finance: This has been supported everywhere mentioned in [5] and [19]; here we list a few of them: [19] introduces four issues in the current state of financial services: “proprietary messaging frameworks, private networks, no standardization and lack of interoperability”. It then points out that XML “makes all these issues easier because, along with web services, or even proprietary vendor development, it can take the data and make it easily accessible and flexible ([19])”.

64
5.1. Reviews on other industries

that “the whole financial document or part of the document can be encrypted using digital signatures” due to their XML formats which “will greatly streamline the confirmation part of the trading process” ([19]). Another supporting statement can be found in [5] that “XML transactions formats allow well-known default values to be easily applied to contracts, and allow the contract details from both sides to be compared automatically ([5])”.

• Biology: The fact that “there are numerous proposals for the use of XML-based standards for representation of information within systems biology” indicates that XML is playing an important role in data exchange in this area. Evaluations in [1] (and also in [21]) shows that XML “is an interesting and easy-to-use format for information representation in biological applications ([33]),” “due to its flexibility, simplicity and interconnection capabilities ([1])”.

• Legislation: [37] clearly states that “the efficiency of managing and processing information in legal documents can be dramatically improved by applying XML techniques”. It is also pointed out in [20] that XML can facilitate both drafting stage and accessibility stage, as well as several other aspects of the legislative process, such as “enforcing some or all the drafting rules defined for our norms”, “fostering easy and sophisticated searching and rendering tools for the public at large”, etc ([20]).

5. Usually more than one XML standards are available in one area:

• Finance: There are nine XML specifications for finance introduced in [5] and [19]—“FIXML, FpML, SWIFTML ([5], [19]), ISO 15022 XML ([19]), XBRL, MDDL, RIXML, NewsML and MarketsML ([5])”—which explicitly verified this statement. [14] also says that “vendors and standards bodies promote different XML standards”; in the meantime more “XML-based standards are emerging, aligned by industry and industry sector, such as retail banking, insurance and investments, and by application, such as payments, research provision and financial reporting.” ([14]).

• Biology: [33] mentions that their “first wide search for XML-based standards within systems biology found 85 standards of varying levels of interest ([33])”, which is indeed a large number. After setting a list of restrictions as well as some criteria, they
5.1. Reviews on other industries

still selected 14 standards for the final review: SBML, PSI MI, BioPAX, CellML, CML, EMBLxml, INSDseq, Seqentry, BSML, HUP-ML, MAGE-ML, mzXML, mzData, AGML ([33]). Three of them (SBML, PSI MI, BioPAX), are also introduced and compared in [32], for a specific sub-area “representing molecular interactions or cellular pathways ([32])”.

- Legislation: [18] lists a set of “current main available European XML standards” for legal documents, some of which are “MetaLex and SDUBWB in the Netherlands, LexDania in Denmark, and the NormelnRete project in Italy ([18])”. Out of the European context, there are more important initiatives such as the “AKOMA NTOSO project aiming at the definition of an XML based document format for legislative and parliamentary documents in African Parliaments” ([18]). Besides, there are also international and “non-proprietary standards” such as LegalXML which is “created for electronic exchange of legal data” ([18]).

6. There is no all-purpose XML standard for each area:

- Finance: This can be implicitly supported in [14] that “standards will realign and be integrated” as “financial service providers continue to converge and reorganize along new business lines”. This demand of realigned and integrated standards indicates that sometimes there is no existing standard being capable enough to support an application or system particularly for new business lines, not to mention an all-purpose one.

- Biology: We have not found explicit supporting statement. However, it is pointed out in [33] that “standards that have been created for a particular and well-defined purpose have often been more popular than general standards” ([33]). Moreover, [33] lists the purposes of all the 14 standards it reviews, which shows that every standard has its own purpose and none of them is designed to target all kinds of applications within systems biology.

- Legislation: It is analyzed in [18] that for the ESTRELLA project, adapting a single existing standard is not able to meet all of their requirements, but to integrate several of them is expected to be ideal (also introduced in 9 below). This indicates that an all-purpose standard that should be able to support this project does not exist.
5.1. **Reviews on other industries**

7. Differences among available XML standards have to be identified:

- **Finance**: [19] introduces the different targets of FIXML and FpML: "FIXML is used to describe information exchanged on standardized instruments while FpML describes information for over the counter derivative instruments ([19])." The various targets of standards, along with the necessity for financial service providers to decide which standards, if any, to use ([19], [14]), result in the demand for decision makers to identify differences between available XML standards in order to choose an appropriate one.

- **Biology**: [33] states that "there is a significant difference in scope between available XML-based formats for systems biology"; even for standards express information on a same, smaller domain "molecular interactions or cellular pathways ([32])", there is still a "large difference in information content in them" ([32]). Therefore, to identify a standard "with the purpose coincides with the user's interest" is crucial for "a user who is in need of a standard for representation" ([33]).

- **Legislation**: It has been illustrated in the ESTRELLA project ([18]) that in order to find and design a standard that fits their requirements, they have studied a set of available standards in different countries and scopes, and had a comprehensive comparison among them. The comparison did help them to extract "the best and most convincing principles" from those reviewed standards, used to develop their own, integrated solution.

8. Identifying differences among XML standards is usually not easy:

- **Finance**: [14] points out that the decision makers, responsible to choose XML-based standards to adopt, are in a "complex, rapidly evolving and difficult-to-navigate landscape ([14])", which indicates that to compare and select suitable ones in such landscape is not an easy job.

- **Biology**: Both [33] and [32] provide comparisons among standards and their comparisons are both based on complex frameworks: [33] first compares them in general including "name, version, defined by, purpose, tools and data". Then it goes to the
5.1. Reviews on other industries

content such as "whether the standards give information on substances, interactions, pathways, compartments, organisms, or experiments" ([33]). In order to compare more thoroughly, [33] also took a "closer look at which kind of information can be defined for each standard ([33])". In terms of [32], which is a paper that particularly focuses on introducing a comparison method, the framework is more formal. It proposes a two-dimension comparison method: "one for comparing the semantic concepts and one for finding information for automatic match between the standards ([32])". Analyzing standards based on these two dimensions are quite complex though, demonstrated by the examples provided in the paper ([32]).

- Legislation: The comparison framework in [18] contains six criteria most of which have several parameters: "architecture and approach of the model (scope of the model, DTD or XML-scheme, prescription versus description, pattern versus free listing of elements, and content model containment versus free content model), structural elements and normative references (content and presentation), metadata (general metadata and metadata inline), temporal elements and versioning (general temporal elements, temporal elements inline and versioning), semantic web and ontology connection (ontology), and special functions (multilingual elements and geographical elements) ([18])". Some extra ones, such as openness and extensibility, are also considered. This comparison framework involves a lot of aspects of those standards, which would require a solid understanding on all of them.

9. Sometimes none of the available XML standards is suitable:

- Finance: [14] states that (which also mentioned in 6 above) "as financial service providers continue to converge and reorganize along new business lines", existing standards will not be capable enough and to realign and integrate standards will be expected ([14]). "This process will also be fueled by providers seeking integration across disparate systems and applications in support of initiatives such as straight-through processing, e-business and customer relationship management ([14])." The necessity of integrating existing standards is also demonstrated by the goal of SWIFT which is to "incorporate FIX and FpML into a new standard referred to as ISO 15022XML ([19])", in order to leverage
5.1. Reviews on other industries

expertise of both pre-trade and post-trade domains ([19]). However, “this is a very slow process and work on the integration has not gone far”, as introduced in [19].

• Biology: [33] mentions the possibility of using multiple standards that “there will be many standards with different scopes existing side by side and the user will often have to cope with handling data in more than one standard ([33])”. To match and translate between standards is difficult though, as “a pure syntactic approach to automatic translation would not be sufficient and that semantic understanding of the standards is important”, stated in ([32]).

• Legislation: With the aim of specifying “an open, general, jurisdiction independent and language independent XML standard to describe legal sources ([18])”, [18] found out that they cannot just “adapt an existing solution to the exclusion of all others” as it is “impractical and presumptuous and severely risks of being ignored by the supporters of all the rejected ones” ([18]). As a result, what they are trying is “to pick and choose the best ideas (or, for that matter, the solutions shared by the most) and try to integrate them, adapting them into a standard that allows each existing standard as a direct sub-construct immediately and automatically obtainable through the use of appropriate applications ([18])”. This analysis clearly shows that for the project ESTRELLA ([18]), all of the existing standards are not suitable, while an integration among them is expected to be the solution.

10. Coordination on standards is necessary:

• Finance: It is agreed in [19] that “the structure of the information exchanged must be in a format that is agreed upon by the communicating parties ([19])”. This is challenging, however, because “a financial service provider does not own, and therefore cannot control, both sides of the data exchange, with the result that the provider cannot unilaterally determine the standards to be used ([14])”.

• Biology: It is pointed out in [1] that “the use of XML as an intermediate medium would be really efficient only if all databases share common or very similar schemas”. Moreover, “whatever language is used, it is always difficult to find an agreement on
5.2. A summary

a common semantics, and when one is found, it is often revised. ([1])

- Legislation: No evidence has been found for this hypothesis, in these three papers.

11. Transformation between XML and relational databases is needed sometimes:

- Finance: [5] lists not only relational databases and XML, but three types of data models in use: database, object and XML. It then points out that “there is a need for tools that make it easier to integrate these data models without any of the three (database, object, XML) looking like an awkward mechanical creation ([5])”; “there is also a need for harmony between how you store your information in a database, how you serialize it for transmission using XML, and how you represent it as objects in memory ([5])”.

- Biology: Although the transformation between XML and relational databases is not mentioned, [1] introduces an idea of joining XML and OODBMS (Object-Oriented DBMS), as it claims that “XML has some weaknesses that match the OODBMS strengths ([1])”. The idea is to use OODBMS to store and retrieve data, and XML to exchange data, so that “queries on the database would then return XML formatted results that could then be exchanged between users, databases or displayed by adequate XML browsers ([1])”. This would, obviously, require a data transformation between XML and OODBMS.

- Legislation: We have not found evidence for this hypothesis in the legal domain, whose reason we believe is that the data related to legislation are always documents, which are usually not stored in relational databases.

5.2 A summary

In this section we provide a summary chart 5.1 explicitly showing if our hypotheses being supported, for each reviewed area, based on our studies introduced in previous chapters/sections. In this chart, "Y" means “having been supported” and “N” denotes “evidence not found”.

70
5.2. A summary

Table 5.1: Summary chart

<table>
<thead>
<tr>
<th>Hypotheses</th>
<th>Civil Engineering</th>
<th>E-Business</th>
<th>Finance</th>
<th>Biology</th>
<th>Legislation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. XML has high expressiveness and flexibility.</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>2. The schema of XML standards is sometimes complex.</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>3. XML is not efficient enough.</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>4. XML standards improve the facility of data exchange among applications.</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>5. Usually more than one XML standards are available in one area.</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>6. There is no all-purpose XML standard for each area.</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>7. Differences among available XML standards have to be identified.</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>8. Identifying differences among XML standards is usually not easy.</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>9. Sometimes none of the available XML standards is suitable.</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>10. Coordination on standards is necessary.</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>11. Transformation between XML and relational databases is needed sometimes.</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
</tbody>
</table>

It is clear shown in the table that all of our hypotheses have been supported by at least three areas. This indicates that our hypotheses are not only fitting to one specific area, but able to be applied in general.
Chapter 6

Conclusion and Future Work

In this thesis, we gave an evaluation on XML standards for actual applications: we first looked at a case study for civil engineering of how to extract information, particularly using XML (Chapter 3). That led us to make some hypotheses on XML standards (Section 1.3). In order to apply our hypotheses more generally, we verified them on several other areas: we gave a comprehensive review on e-business (Chapter 4), and a broad study for finance, biology and legislation (Chapter 5).

Our hypotheses, which have been verified to be supported generally, show that XML standards are playing an important role today: they are widely used and improve the facility of information exchange. However, they are still in infant stage and hard to use. A lot of challenges exist when using XML standards in reality:

- expert knowledge on XML is needed to understand, adopt, develop or coordinate XML standards due to the complexity and diversity of their schemas;
- solid understanding and comprehensive evaluations on a variety of potential standards are required to make the decision from multiple choices;
- besides, the scalability problem limits the performance and usage of XML standards.

All these challenges imply that we, as the data management research community, need to think of how to help XML become the more useful medium of data exchange that the world needs it to be. Here, we propose a few ideas as future work:

- Developing mature tools for both XML and XML standards: Current XML tools provide functions such as tree views and schema generating. However, it is still hard to understand the syntax and semantics of an XML file when its schema is complex, particularly for non XML-experts. There are also editors developed for specific
standardstomakeeasiertheusageofthosestandards(forexample,thoneintroducedin[37]),butalargenumberofotherstandardshavenosuchsupportingttools.WeeexpectthatmaturetoolsforvisualizingandeditingXMLandXMLstandardswillenableuserstounderstand,compareandadoptXMLstandardsmoreeasily,withoutbeinganXMLexpertorITspecialist.

•Optimization:AlthoughthetextnatureofXMLcannotbetchanged,optimizationsonparsing,querying,aswellastransferringXMLformatteddataareallpotentialwaystosolvetheefficiencyproblem.

•Standardizationforstandarddefining:Wedonotexpectall-purposestandardsbeingcreated,thereforetheissueofstandardschoosingandcoordinatingwillalwaysexistforapplicationdevelopers.Currenthallengesonitarepartlycausedbynon-harmonizedmethodologies,suchasvarioussemanticsanddatastructures,usedonstandardsinonearea.Standardizationonstandarddefiningisexpectedtobea
tawaytoeasetheusageincludingcomparison,selectionandcoordinationofXMLstandards,andeffortshavealreadybeenmadeonsomeareas(suchas[31]).Thisisalsochallenging,however,asitwillresultin
changesonexistingstandardsthathavebeenwidelyadoptedalready.
Bibliography


Chapter 6. Bibliography


76