CONSTRUCTION SMART FORMS: AN APPLICATION OF INFORMATION TECHNOLOGY TO REDUCE WASTE BY INCREASING INTEROPERABILITY

A Thesis Presented by

Babak Jalalzadeh Fard

to The Department of Civil & Environmental Engineering

in partial fulfillment of the requirements for the degree of

Master of Science in Civil and Environmental Engineering

In the field of Construction Engineering & Management

Northeastern University Boston, Massachusetts

January 2011
Acknowledgements

First, I would like to thank Professor Ali Touran for his support and guidance during my graduate studies. To Professor Daniel M. Dulaski, I give thanks for his review and valuable comments on my thesis. Finally, I would like to thank the faculty and staff at Northeastern University for their instruction and guidance during my graduate studies.

And I wish to express my gratitude to my beloved family for their unconditional support and endless love they have given me my entire life.
Abstract

This subject of this thesis is Construction Smart Forms. The idea is based on the fact that a large amount of data in a construction project is alphanumerical and is gathered by fillable forms. Building Information Modeling (BIM) has made major improvements in data interoperability for geometrical data and 3D models. But for expanding its reach to contain all information shared in a construction project during its lifetime, and beyond, it needs to provide standards and solutions for all the data shared in the projects. eXtensible Markup Language (XML) is the most recognized tool for data sharing in Information Technology (IT) industry and must be used alongside Industry Foundation Classes (IFC) to empower BIM for reaching its full potential. There are efforts underway to develop XML schemas as global standards for data exchanged in construction projects. The thesis is about using appropriate applications and methods that were selected from among different available solutions for developing the Smart Form system. A prototype system was implemented on data in a set of fillable forms that were used in selecting the best project delivery method.

A software system was developed by the author to make a sample template of the Smart Form system. This system is a collaboration of these technologies: pdf dynamic forms, XML, Java programming language, Document Object Model (DOM) technology, and a database system. These technologies are explained very briefly through the text to give the reader a brief insight to the topic and might be used as a starting point for learning about each issue.

The goal of this thesis and the developed software system is to overcome the following inefficiencies in the traditional way of using paper fillable forms:

- To reduce the consumption of paper, resulting in saving of natural resources
• To reduce the time and human resources which are used for entering data into different systems, in traditional method

• To address the probability of error in data entry by providing a solution for data sharing and automatic data transfer
# Table of Contents

CHAPTER 1. Introduction ........................................................................................................ 9  
1.1. Overview .................................................................................................................. 9  
1.2. Building Information Modelling (BIM) ..................................................................... 11  
1.2.1. Building Blocks of BIM .................................................................................... 14  
1.3. Motivation ............................................................................................................... 16  
1.4. Objectives of the Research .................................................................................... 17  
1.5. Organization of the Thesis .................................................................................... 18  

CHAPTER 2. Explanation of the Forms for Delivery Selection System............................... 21  
2.1. Project Delivery Methods Used in ACRP Report 21 ............................................. 22  
2.1.1. Design-Bid-Build (DBB) ................................................................................. 22  
2.1.2. Construction Manager-at-Risk (CMR) ........................................................... 22  
2.1.3. Design-Build .................................................................................................... 23  
2.3. The Electronic Version of ACRP Report 21 Forms .............................................. 24  
2.3.1. Form 1: General Specifications ..................................................................... 25  
2.3.2. Form 2: Analytical Approach ........................................................................ 26  
2.3.3. Form 3: Weighted Matrix Approach ............................................................... 28  
2.4. Summary ............................................................................................................... 30  

CHAPTER 3. XML Technology- A Standard Carrier of Data .............................................. 31  
3.1. What is XML ......................................................................................................... 31  
3.2. A Brief Look Into XML ....................................................................................... 35  
3.2.1. The Syntax of an XML Document ................................................................. 35  
3.2.2. Some Important Features and Related XML Technologies ......................... 36  
3.3. Some of the Most Recognized XML Schemas in Construction Industry .......... 39  
3.4. The Document Object Model (DOM) .................................................................. 46  
3.5. Web Services Definition Language (WSDL) ....................................................... 48  
3.6 Summary ............................................................................................................... 49  

CHAPTER 4. Database Design Principles ............................................................................ 50  
4.1. Preliminary Steps for Designing a Database ....................................................... 50  
4.3. Business Analysis for Developing Database Solutions ..................................... 53  
4.4. Database or Spreadsheet? ................................................................................... 53  
4.5. Database Concepts ............................................................................................... 55  
4.5.1. Lists and Related Tables ................................................................................. 56  
4.5.2. Relational Models ........................................................................................... 58
List of Figures

Figure 1.1. Change Of Productivity in Construction Industry Compared to Non-Farm Industry (McGrawHill Construction, 2009) ................................................................. 10
Figure 1.2. The View Above Shows the Pipe Created Manually with the Gate Valve Clashing with the Structural Beam. ................................................................. 12
Figure 1.3. Some Common Connotations of Multiple BIM Terms (Succar, 2008) .... 13
Figure 1.4. Three Stages of BIM Maturity (Succar, 2008) ................................. 14
Figure 1.5. The Fundamental Classes of IFC (Wix, 2007) ................................. 15
Figure 1.6. The Process for Defining End User BIM Data Exchange (See, Fall 2008) ................................................................. 16
Figure 2.1. The Go/No Go Start Screen. By Pressing Finalize Button the Form Will Produce the Required Tables for Tier1 ................................................................. 27
Figure 2.2. If the User Considers DB, This Table Will Let the User Select the Appropriate DB Method(s) ......................................................... 27
Figure 2.3. The Template for Issue No. 1 (Project Size/Complexity) for A Situation Where DBB and CMR Methods Are Selected for Tier1 ............................................. 28
Figure 2.4. A Filled-Out Weighted Matrix Form. Refresh Button is Used to Return All 4 PDM Choices in the Table When Some are Removed by the User ................. 29
Figure 3.1. XML Documents Can Act as a Connection Point for Many Programs and the Worldwide Web ................................................................. 34
Figure 3.2. The Process, Showed by Dash Lines, Starts with A Language for Model Definition, and Many Instances are Made from the Model (Liebich, 2002) .............. 41
Figure 3.3. Comparison of Data Definition in IFC and Ifcxml. (adapted from (Nisbet & Liebich, 2007)) ................................................................. 42
Figure 3.4. The Anticipated Role of SPF and Ifcxml (Nisbet & Liebich, 2007) ...... 43
Figure 3.5. XML DOM Tree of Book Example ................................................. 48
Figure 4.1. Overview of Exchange Standard Development and Use Process as Defined In NBIM Standard Version 1, Part 1 ........................................................................... 52
Figure 4.2. List of Materials with Their Omniclass Numbering System .......... 56
Figure 4.3. List of Materials and the Providers .................................................. 57
Figure 4.4. Modification Problems in Material and Merchant List .................... 58
Figure 4.5. The Relational Model of MERCHANT and MATERIAL Tables ......... 59
CHAPTER 1. INTRODUCTION

1.1. OVERVIEW

The application of Information Technology (IT) in all aspects of life has transformed business, industry and engineering in recent years. Different industries have benefited from IT in various ways. The construction industry has used IT in managing contracts, estimating and scheduling, resource management, bidding, cash flow management, project control, cost estimating and more (Hegazy, 2002). Despite the IT revolution, these and other technological improvements have not been able to stop the decline in productivity in the construction industry. Figure 1.1 shows the changes in productivity in Construction industry between 1964 and 2004, while at the same time this productivity has been rising more than 200% in non-farm industries such as manufacturing (McGrawHill Construction, 2009).
Part of the inefficiencies in the construction industry is attributable to the consumption of natural resources. A large proportion of world’s raw material is consumed by the construction industry. 40% of these resources are used in buildings. Facilities are the users of 65.2% of world's energy. Adding these statistics to the fact that construction is a highly inefficient industry with 57% waste, reveals the fact that it has a major responsibility in energy, environmental, and natural resources crises. This waste represents $600 Billion out of $1.288 trillion construction industry in the United States in 2008 (buildingSMART alliance, 2007). Considering that this is just the monetary value of direct waste without considering the long-term effects on natural and energy resources, the growing crisis needs immediate consideration.

One source of ineffectiveness is the old business processes which depend on paper base data exchange which may include retyping the same data again and again. Information Technology has provided a unique opportunity for different businesses to develop new processes which reduces the waste in time and resources and improves
interoperability among different software and stakeholders in a project. One recent technological improvement is the concept of Building Information Modeling (BIM), a robust electronic medium for infrastructure, which has the potential to impact all aspects of a facility’s lifecycle.

1.2. BUILDING INFORMATION MODELLING (BIM)

One of the most recent advancements in application of IT in Architectural, Engineering and Construction (AEC) industry is BIM. There are many different definitions available for BIM. To fully understand the full potential of BIM, it is very important to understand its different aspects and definitions. For the purpose of this research and considering BIM as a concept and using different resources, this description is presented by the author of this thesis, to cover different aspects and viewpoints:

BIM is the concept of creating and incorporating processes, open information and interoperability standards in Architectural, Engineering, Construction, Owner Operator (AECOO) industry to provide repeatable, verifiable, transparent, and sustainable information throughout the building lifecycle and to facilitate digital collaborative software tools for all stakeholders during the facility’s lifecycle. Over the last 10 years the concept of merging CAD(Computer Aided Design) and data management requirements for extending the shelf life of information and repurposing previously collected information, helped to create what we now know to be BIM software (Hecht, 2010).

In the following, some important benefits of BIM are introduced:
- Clash detection, which makes it possible to identify possible points of clashes, such as duct passing through a column, in the design phase. (Figure 1.2)

Figure 1.2. The View Above Shows the Pipe Created Manually with the Gate Valve Clashing with the Structural Beam.

- Reducing re-entering the same data by providing a single repository for data related to a project and eliminating re-entering the same data in different applications and by different individuals.

- Reducing conflicts and changes during construction by providing a complete three dimensional (3D) virtual model containing all the geometrical details of the project, before starting the construction phase.

- Fewer Request For Information (RFI) and field coordination problems.

---

- Interoperability between different software and stakeholders.

- Increasing the speed of project execution.

- Analyzing different scenarios which is made possible by automatic data transfer between different software applications in a project.

Other benefits can be added to this list. Figure 1.3 describes some important connotations of Building Information Modeling.

![Image of figure 1.3](image.png)

**Figure 1.3. Some Common Connotations of Multiple BIM Terms (Succar, 2008)**

The incorporation of BIM is still not complete to take all the possible advantages. From a Pre-BIM approach which follows the traditional method of exchanging documents and drawings, to the full integration of BIM, three basic stages can be considered as shown in Figure 1.4 (Succar, 2008).

Stage 1: object-based modeling: users generate single-disciplinary models within design, construction or operation.

Stage 2: model-based collaboration: Having developed single-disciplinary modeling expertise in Stage 1 implementations, Stage 2 players actively collaborate with other disciplinary players. The interchange of models between different software and different file formats is an example of this collaboration.
Stage 3: network-based integration: BIM models in this stage are interdisciplinary models connected to scheduling, cost and other data. These models require major reconsideration of contractual relationships, risk-allocation models and procedural flows.

![Figure 1.4. Three Stages of BIM Maturity (Succar, 2008)](image)

1.2.1. Building Blocks of BIM

Industry Foundation Classes (IFC) is the core technology behind BIM. The IFC specification is a neutral data format to describe, exchange, and share information typically used within the building and facility management industry sector (AEC/FM) (buildingSMART, 2010). It was introduced as the “common language for interoperability in the building industry” at the 1995 AEC Systems conference in Atlanta. The idea for developing IFC was inspired from the role that STEP (the Standard for the Exchange of Product Model Data) had played in the manufacturing industry, from around 1985 onward. STEP uses the Object Oriented paradigm to solve some complex problems in manufacturing industry by defining object oriented data models. The Object Oriented paradigm is a prominent software development paradigm that puts the data and its related functions into one entity called object. Both IFC and STEP are object oriented data schemas based on EXPRESS data modeling language. IFC defines a hierarchical set of objects which can hold data for geometry, calculation, quantities, facility management, pricing, etc for many different
professions including Architects, technical consultants, contractors, property owners and the materials industry.

All classes in IFC are derived from *ifcRoot* class. Three classes are immediate derivates of *ifcRoot*. They are *ifcObject*, *ifcPropertyDefinition*, and *ifcRelationship*. All the objects in the AEC/FM industry are defined based on *ifcObjects*. Some examples are *ifcWall*, *ifcDoor*, etc. *ifcPropertyDefinition* is the root for defining the properties of different objects, and *ifcRelationship* is the super class for the relationships between the objects. Other classes derived from these basic classes can model nearly all products, situations, business processes and other aspects of AECOO industry. More details can be found in the technical section of BuildingSMART website ([www.buildingsmart.com](http://www.buildingsmart.com)).

IFC is so large that no single software application can put all of its facets into use, except model servers. In fact, each application needs some parts of IFC which are relevant. For example, a structural analysis program only needs to work with the structural objects such as beam and column, and the related structural properties.

![Diagram of IFC Classes](image)

*Figure 1.5. The Fundamental Classes of IFC (Wix, 2007)*

In the summer and fall of 2007, an integrated process for defining end user BIM data exchange requirements was developed by the US National BIM Standard (NBIMS) committee, IAI/buildingSMART International, and several organizations using IFC based BIM data exchange. The name of this integrated system is the *IFC Solutions*
Factory. It includes the following four sub-processes, toolsets, and templates (Figure 1.6) (See, Fall 2008).

![IFC Solutions Factory](image)

**Figure 1.6. The Process for Defining End User BIM Data Exchange (See, Fall 2008)**

At the time of writing this thesis (Dec 2010), the IFC 2x4 Release Candidate 2 is available. IFC is used in major BIM software as the common standard for exchanging data.

### 1.3. Motivation

In spite of growing number of companies and agencies in AECOO industry which gravitate to BIM, for many users its application is limited to using some widely available 3D modeling commercial software which are equipped with scheduling and quantity takeoff capabilities, known as 4D and 5D, respectively. The generally known BIM software deals with geometrical and functional information of facilities in a construction project.
The reason for this look is that BIM development was started by adding data management capabilities to CAD software. Transactional data contain a large number of data exchanged in AECOO, and other standards and methods for containing this large set of data are being added to the scope of BIM. For example, COBie (Construction Operations Building Information Exchange) makes it possible to enter the data as they are created during design, construction, and commissioning (National Institute of Building Sciences, 2010). This information is essential to support the operations, maintenance, and the management of the facilities assets by the owner and/or property manager. In its most direct and usual form of use, COBie is a formatted Excel spreadsheet which can be populated during the whole lifecycle of a project. COBie data are importable into IFC file format and are based on a simple and widely used solution. It uses structured and standard Excel sheets to organize data in a standard way.

Having the correct understanding about BIM’s identity and products is very important for everyone who is trying to develop IT solutions for AECOO industry.

1.4. Objectives of the Research

The subject of this thesis is based on the fact that traditional paper forms play an important role in capturing some data during the lifecycle of an AECOO project, while they are inefficient. The main goal of this thesis is to develop and test some aspects of “Construction Smart Forms”. The Construction Smart Forms aim to improve the way data are gathered by traditional forms and reduce the inefficiencies and the duplication of data in a project. These forms are defined as fillable pdf forms which can contain many different aspects starting from seamless data transfer from
the form into the destination, checking different rules and regulations where applicable, and many different features which can make a form Smart. The idea of naming these forms as Smart comes from the name buildingSMART which is the main international organization which develops BIM standards, and the smart way of manipulating and transferring data in the system. Smart Forms can be considered as a system which gets data from the user by pdf fillable forms and transfers them to the final destination. This may contain different IT solutions. The system developed for this thesis contains these parts:

- A set of three pdf dynamic forms which are described in more detail in Chapter 2.
- XML (eXtensible Markup Language) file exported from the filled forms and contains data provided by user.
- A set of three modules of Java codes which transforms the XML exported document to a document which can be imported into the database.
- A database which stores the data gathered by the pdf forms.

Smart forms can reduce the rework for entering the data again and again in different applications and by different people. It is also an important component of a document management system and in harmonization with traditional BIM tools for managing all data in an AECOO project. Data will also be automatically stored in one central data repository and are easily manageable.

1.5. ORGANIZATION OF THE THESIS

Chapter 2 describes the purpose and functionality and of the forms which were used as a case study for developing a sample of the Smart Forms. A brief introduction into
the project delivery methods is presented which are used in the forms and their related report.

eXtensible Markup Language (XML) is considered as the career of data for the Smart Form system. It is a widely used standard way of defining data structures which is supported by web and most of software application. Chapter 3 is about XML. It introduces XML and explains why it is so important in IT and web-based tool; and as a result, the reason it is gaining importance in the application of IT in AECOO industry. Then it introduces the syntax of an XML document and explains important features and related XML technologies such as Namespaces and XML Schemas. Some of the most important XML schemas for Construction industry are introduced. Document Object Model (DOM) is then introduced as a way to manipulate XML documents, which was used in the Java code developed for this project. The chapter concludes with a brief insight into Web Services Definition Language (WSDL). Although, not used in this project, familiarity with WSDL, as a very powerful tool which will be used more in the future, will be beneficial.

Construction Smart Form system should also contain a solution for storing data gathered by the form into a central data repository. For this thesis a database system was considered for this purpose and a database was developed based on the database concepts which are the subject of Chapter 4. Chapter 4 introduces Lists and related tables and covers the anomalies that happen in lists. Then it talks about data modeling, Normalization, and entity-relationship models. The meaning and the application of key fields are explained and Entity-Relationship models are introduced with an introduction to the concept of Cardinality. The chapter contains short examples for each subject.
Construction Smart Forms are supposed to be user friendly and change their size and functions based on the preferences of the form user. These features can be made possible by applying the principles of dynamic forms to the ordinary pdf forms. Chapter 5 explains different aspects of dynamic forms and introduces LiveCycle Designer™ as the tool for making these kinds of pdf forms. The case study developed for this thesis uses dynamic pdf forms to provide more manageable and flexible forms which can fit different situations. The main tools for making a dynamic form are Sub forms and Dynamic Tables, which are described.

A Smart Form system template was developed for this thesis to make a sample of the Construction Smart Form. Chapter 6 explains the system’s components and how they are designed and work together. It first introduces the principles which should be considered in designing forms. Then, different parts which make the system work are described. This explanation starts with the criteria which can help in selecting between spreadsheets and databases as the repository for data. Then it explains the method which is used in this system to transfer data. Data gathered in the form is exported into XML document, which is then changed into a XML document importable to the database by a Java Code which uses DOM technology.

The developed system was then validated by entering some sample data into the forms and performing the processes required to transfer the data to the database. Appendix B contains the Java codes which were developed for the system and a sample of System application for the data gathered in the first form and then exported into an XML document. Then it shows the three step changes to the XML document which makes the XML document ready for import into the database.
CHAPTER 2. EXPLANATION OF THE FORMS FOR DELIVERY SELECTION SYSTEM

In order to show the application of smart forms in construction, a set of forms that were originally developed by the author of this thesis for a project funded by the Airport Cooperative Research Program (ACRP) of the National Academy, were selected as to be improved into a Smart Form system. The original forms were published by the National Academy as ACRP Report 21 (Touran, et al. 2009). The methods and the system developed in this thesis could use any other form used for different purposes in a facility’s lifecycle. These forms were selected as a ready set of dynamic pdf forms, also to provide a system for gathering data about the trends in selecting the project delivery method, for future research. In the following sections, construction project delivery methods, ACRP Report 21 and the purpose of its forms are introduced.

The mentioned forms are electronic versions of the tables and forms presented in ACRP Report 21: A Guidebook for Selecting Airport Capital Project Delivery Methods (Touran, et al 2009). The forms contain in electronic format Appendices C, D, and E of Report 21. The forms can be viewed and filled using Acrobat Reader™ which is freely available to all users. The original Report 21 documents the benefits and disadvantages of various project delivery methods for capital airport projects and provides guidelines for selecting an appropriate delivery method for a specific project.
2.1. PROJECT DELIVERY METHODS USED IN ACRP REPORT 21

Project delivery methods considered in ACRP Report 21 are Design-Bid-Build (DBB), Construction Manager-at-Risk (CMR), and Design-Build (DB). In the following sections, these delivery systems are briefly described.

2.1.1. Design-Bid-Build (DBB)

This is the traditional method used for different construction processes. In this method, the design is usually completed under a design contract, and the constructor is usually selected through a bidding process; the contract usually goes to the lowest bidder. Sometimes, usually in privately funded projects, the contractor may be selected by negotiations (Ghavamifar, 2009). In DBB, each stage of the project starts after the completion of the previous stage. For example, the construction must start after full completion of the design, and after awarding the contract to a contractor (America, 2004).

2.1.2. Construction Manager-at-Risk (CMR)

CMR projects contain a contract between the owner and a construction manager who accepts the responsibility of managing the project to be delivered on time and not over the budget. Usually, the owner takes the responsibility for providing the design documents and works with the designer. CMR may start during the design phase and provides preconstruction services to the owner before engaging in the construction process. Typically, in CMR contracts the owner agrees to pay up to a Guaranteed Maximum Price (GMP) and the costs more than that agreed price are the
responsibility of the CM. Cost savings are usually shared between the owner and the CM.

2.1.3. Design-Build

In this delivery method both design and construction are contained in the same contract awarded to a single entity which is called Design-Builder. The design-builder is usually selected by one of two methods: One-step and two-step process. The one-step process provides for competitive evaluation of technical proposals, with the contract award decision based on best value (BV) to the owner agency. This is the method which is summarized as DB (BV) in the forms. In the two-step method, the owner develops a RFQ (Request For Proposal) or RFP (Request For Performance) that describes the essential project requirements in performance terms. After evaluating the proposals, the owner selects the candidates for the next step which leads to contracting the process. This method is known as the Qualification Based System (QBS) and is shown as DB (QBS) in the ACRP Report 21 forms. In the DB method, the DB entity is liable for all design and construction costs and the contract is usually a fixed-priced. One of the best benefits of this method is its capability of shortening the duration of a project by overlapping design and construction, because one entity is responsible for both (Ghavamifar, 2009).

2.2. The Functionality of the Forms

A two-tiered project delivery selection framework is developed that can help the owners evaluate benefits and disadvantages of each delivery method and select the most appropriate one for their project. Tier 1 is a qualitative approach that allows the
user to document advantages and disadvantages of each competing delivery method. The user can then review the results of this analysis and select the best delivery method. If at the conclusion of this analysis, a clear option does not emerge, the user then moves on to Tier 2. Tier 2 is a weighted matrix approach that allows the user to quantify the effectiveness of competing delivery methods and select the approach based on quantitative analysis. The process produces a Project Delivery Decision Report that documents the decision-making process and details relevant decision factors to fine-tune future project delivery decisions.

2.3. THE ELECTRONIC VERSION OF ACRP REPORT 21 FORMS

The purpose of the electronic companion is to facilitate the process of evaluating competing delivery methods for the owner’s team. The owner’s team can use this tool to go through various steps envisaged in Tier 1 and Tier 2, document project goals, and evaluate various pertinent issues that affect the delivery method, and select the most appropriate delivery method. It will provide the user the ability to obtain relevant reports and perform sensitivity analysis to assess the effect of changing the responses to the questions to the choice of delivery method. Communication between airport’s team members will be facilitated as the results of the analysis can be transmitted using electronic mail. Overall, the current electronic tool is a major enhancement to the PDM decision support system already developed.

The procedure for using the forms is simple. The owner’s team goes through the three files listed under Part 1, Part 2, and Part 3. In each step, the required information is entered into the forms. The forms can be saved and shared between analysts and decision makers and serve as an archival document for the project. The electronic forms were developed using Dynamic form by LiveCycle Designer™, and are now
freely available in the related ACRP webpage in the following address:

<http://www.trb.org/Main/Blurbs/A_Guidebook_for_Selecting_Airport_Capital_Projects_162449.aspx>

In the following, the forms are described individually.

2.3.1. Form1: General Specifications

A general description and characteristics of the project are entered into this form. The form contains multiline expandable text boxes which can roll into the next page. This form covers the first step in the project delivery system selection approach described in ACRP Report 21. The Guidebook introduces the following items as important information that should be covered in the project description:

- Project Name
- Location
- Major Features of Work
- Estimated Project Budget
- Estimated Project Delivery Period
- Required Delivery Date (if applicable)
- Source(s) of Project Funding
- Project Site Dimensions or Project Limits
- Security Issues or Concerns
- Rate of Return on Capital Investment/Payback Period (if applicable)
- Major Schedule Milestones
- Major Project Stakeholders
- Labor Union Status
- Major Challenges (as applicable)
- Main Identified Sources of Risk
- Sustainable Design and Construction Requirements

Some of the above items may need several lines which are provided by using dynamic features in the form. Also the number of rows in the table of milestones can be set to the required number.

2.3.2. Form 2: Analytical Approach

This form covers Tier 1-Analytical Approach in ACRP Report 21 (Touran, et al 2009). The process starts with the Go/No Go selection table (Figure 2.1). A delivery system will be considered for Tier 1 process if the applicable option is selected for all of its related criteria. If DB method is selected, the table in Figure 2.2 appears on the form to allow the user to consider one or both DB methods in the analytical approach. By pressing the Finalize Go/No Go button new pages will be added to the form which contain the advantages/disadvantages tables and summaries for the 19 issues used in analytical approach. The tables are resized to contain the selected methods, only.
Figure 2.1. The Go/No Go Start Screen. By Pressing Finalize Button the Form Will Produce the Required Tables for Tier1

Figure 2.2. If the User Considers DB, This Table Will Let the User Select the Appropriate DB Method(s).

Figure 2.3 shows a sample template of Tier 1 (Analytical Approach). The user rates the delivery method as an advantage or disadvantage for dealing with each statement and also rates each delivery method with regard to that issue in the summary part at the bottom of the template. A summary table is automatically generated after the user goes through the 19 issues and rates them vis-à-vis delivery methods. The forms provide user’s help. Each item shaded in blue is a hypertext item. Pressing each blue shaded item will provide reference to the original report where the item of interest is explained in sufficient detail.
### Issue 1: Project Size/Complexity

#### Design-Bid-Build (DBB)

<table>
<thead>
<tr>
<th>Issue Statements</th>
<th>Advantage</th>
<th>Disadvantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBB has been shown to work on projects of all sizes and levels of complexity, but the research case studies found that airports tend to select DBB on smaller projects.</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>As projects grow in size and complexity, the amount of owner staffing required to oversee DBB can become very large.</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

#### Construction Manager at Risk (CMR)

<table>
<thead>
<tr>
<th>Issue Statements</th>
<th>Advantage</th>
<th>Disadvantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMR has been shown to work on projects of all sizes and levels of complexity, but the research case studies found that airports tend to select CMR on larger and more complex projects.</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>On projects of large size and complexity, CMR can use multiple bid packages to optimize responses from proposers, but this approach results in more complexity in management.</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

#### Summary

<table>
<thead>
<tr>
<th>1. Project Size/Complexity</th>
<th>DBB</th>
<th>CMR</th>
</tr>
</thead>
</table>

| Key: | | |
| 1. Most appropriate delivery method |
| 2. Appropriate delivery method |
| 3. Least appropriate delivery method |

**Comments:** The project size could be prohibitive for DBB on this project.

---

Figure 2.3. The Template for Issue No. 1 (Project Size/Complexity) for A Situation Where DBB and CMR Methods Are Selected for Tier1

### 2.3.3. Form 3: Weighted Matrix Approach

This form is used for Tier2-Weighted Matrix Approach which is described in Chapter 5 of ACRP Report 21 (Touran, et al 2009). The user enters the selection factors and their descriptions in the first table. The number of selection factors is limited to ten and their titles are automatically transferred to the Weighted Matrix table (Figure 2.4). In the Weighted Matrix table, user inputs Factor Weights and Scores for each selection factor and the software calculates total scores for each delivery method.
Different validation checks are applied in the form for controlling the rules which should be considered for the values entered. In case of a breach of a rule the appropriate warning will be shown. These rules are as follows:

- The sum of factor weights must total 100. A warning is present at the bottom of the table which disappears when this summation reaches 100. The form does not calculate the total scores for the delivery methods, if the sum of factor weights is anything but 100.
- Each score must be between 1 and 10.

Figure 2.4 shows a filled-out weighted matrix form. The user can remove one or more delivery methods from the table by pushing the appropriate “Remove” button at the top; by pushing “Refresh” button the table will return to its full situation consisting of four PDMs.

<table>
<thead>
<tr>
<th>Selection Factors</th>
<th>DBB</th>
<th>CMR</th>
<th>DB (QBS)</th>
<th>DB (BV)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factor Weight</td>
<td>Score</td>
<td>Weighted Score</td>
<td>Score</td>
<td>Weighted Score</td>
</tr>
<tr>
<td>Project Size</td>
<td>24</td>
<td>5</td>
<td>120</td>
<td>3</td>
</tr>
<tr>
<td>Limited Time</td>
<td>37</td>
<td>8</td>
<td>296</td>
<td>6</td>
</tr>
<tr>
<td>Limited Budget</td>
<td>23</td>
<td>9</td>
<td>207</td>
<td>4</td>
</tr>
<tr>
<td>Security</td>
<td>16</td>
<td>3</td>
<td>48</td>
<td>7</td>
</tr>
<tr>
<td>Total Score</td>
<td>100</td>
<td>671</td>
<td>498</td>
<td>521</td>
</tr>
</tbody>
</table>

Figure 2.4. A Filled-Out Weighted Matrix Form. Refresh Button is Used to Return All 4 PDM Choices in the Table When Some are Removed by the User.
2.4. SUMMARY

This chapter provided a brief explanation about the functionality of a set of dynamic forms which were originally developed by the author of the thesis, and were used as a case study to test the concepts of Construction Smart Forms.
CHAPTER 3. XML TECHNOLOGY - A STANDARD CARRIER OF DATA

3.1. WHAT IS XML

XML stands for eXtensible Markup Language. It can be considered as a standard, application independent tool for storing and exchanging data. XML is mostly used for data sharing and web developments, and is considered as an important part of the Construction Smart Form, the subject of this thesis. For the purpose of the system developed for this thesis, it is used as a carrier for data between the fillable form and the database.

The World Wide Web and different applications working within it have an interactive way of exchanging data which can be called message-based data sharing. Comparing to file data sharing which puts all the data in a file, in this form only the part of data that fulfils the requirement is provided which is best delivered by xml formats. It provides a standard structure for data to be transferred and manipulated independent of the application they are used in and their visual characteristics. The same data might be viewed in different shapes or be used in various applications for different purposes. The World Wide Web Consortium (W3C), as the main source for XML, defines it as:

“The Extensible Markup Language (XML) is a simple text-based format for representing structured information: documents, data, configuration, books,
transactions, invoices, and much more. It was derived from an older standard format called SGML (ISO 8879), in order to be more suitable for Web use.”

XML is also known as the standard way for sharing and formatting the data on the World Wide Web. XML defines a generic syntax to mark up data with simple, human readable format. It provides a flexible format that can be widely used in different programs and over the web. In its syntax, XML is very similar to HTML (HyperText Markup Language) but it doesn't have any predefined elements. This high flexibility allows the developers to define their own sets of data structures. In this way, a structural designer for example, can define standard data structures for representing the analytical elements and material properties, a construction professional can make a data structure to create a standard way for storing project information, and the same for every kind of business. XML is flexible in defining new elements, while it is very strict about how to define the elements and structure of a data model and provides a united grammar for all XML documents. This set of standard rules and grammar, makes XML parsers capable of reading the XML documents into many different programs. There are some general purpose XML parsers that can read the tree structure of any XML document and extract its components like tags, elements, attributes and so on. Many software programs have their own specified XML parser that can read and write their XML documents and decipher them to meaningful data for that software. Most major software applications have XML definitions for their elements and are able to export XML files and also contain XML parsers to import their XML files (Rusty & Means, 2001). It must be noticed that XML documents cannot perform or execute any tasks by them. They just define data structures or contain the required data.
The use of XML technology is increasing in Architecture, Engineering and Construction (AEC) industry and Facility Management (FM) systems. Architecture, Engineering, Construction, Owner Operator Phase 1 (AECOO-1) Joint Testbed was a collaborative effort, led by buildingSMARTalliance (bsa) and Open Geospatial Consortium (OGC), and the participation of a group of AEC firms to develop methods for advancing interoperability of BIM, and was specifically applied to quantity takeoff and energy analysis domains. The summary report of AECOO-1 testbed suggests that incorporating XML technology into BIM must be considered (Hecht, 2010). Construction projects produce and exchange a huge amount of information during their lifecycle and seem to be an appropriate environment for using XML’s potential capabilities. During the lifecycle of a construction project, there are many situations when a segment of data is needed to be exchanged between the parties in different forms such as spreadsheets, CAD files, and many more. The traditional BIM systems, IFC files, rely on file based data sharing, and are not facilitated to work on the internet and message based data exchanges. XML is the widely accepted and used IT standard for message based data sharing which is supported by an unlimited number of applications. Also, creating XML files can be done with any text editor which can save them in plain text; XML files are in ASCII (American Standard Code for Information Interchange) format, readable by all text editors, although there are some tools for developing more complicated XML documents. Furthermore, most programs have an option for exporting the data and their environment into XML files.

Different sets of XML data structures can be defined to transform information in a construction project. For example the delay of an activity is calculated in Primavera P3, and then it may appear in a different environment such as reports in MS Word format, Excel sheets and tables, in a PowerPoint presentation which shows the current
situation of the project, and many other programs which might be used by different groups involved in the project. XML makes data available to all these programs at one point, dramatically reducing the time spent and errors which may happen in traditional data transfer methods and solving the problem of multi sources for information. Figure 3.1 is a schematic view of the idea of having one source of XML data that can be used by different programs and uses.

Figure 3.1. XML Documents Can Act as a Connection Point for Many Programs and the Worldwide Web

XML can simplify data sharing, data transport, platform changes and make data more readily available. New internet languages have been developed and are being developed by XML for various purposes (W3Schools, 2010). Some of the more well known are:
XHTML is the extended form of HTML, the language which web pages are written by.

- WSDL (Web Services Definition Language) for providing web services
- WAP (Wireless Application Protocol) and WML (Wireless Markup Language) for accessing web from a mobile phone
- RSS (Really Simple Syndication) for news feed
- And many more

3.2. A BRIEF LOOK INTO XML

In the following section, the structure of an XML document and the related important technologies will be introduced briefly.

3.2.1. The Syntax of an XML Document

Like any other data definition language XML has its own syntax for defining the structure and the components of a data set. Having an understanding about this syntax will help the reader to understand xml documents. Here you can see a simple XML document which contains data about a book.

<?xml version="1.0"?>
<Book>
  <Title>XML: Visual QuickStart Guide</Title>
  <Author>Kevin Howard Goldberg</Author>
  <Publisher>Peachpit Press</Publisher>
</Book>

The first line introduces the document to the parser. It includes the version of the corresponding XML and in some cases it may have encoding information. In the
following, <Book> tag indicates the start of the root element. Every XML document must have a root element that all other information are kept between its start and end tags. Here, </Book> is the end tag which shows the end of the Book element. XML documents form a tree structure which starts from the root and each element can contain unlimited number of elements known as children. The “children” elements in an XML document must be nested within their parent and no nest can cut another one. The children elements can also have unlimited children elements. Also, the tags are case sensitive. For example, if </book> was used as the end tag instead of </Book>, it would not have worked properly.

These mentioned simple rules are basic considerations in making a well formed XML document. In the following, some important terms of XML and a few related technologies are introduced.

3.2.2. Some Important Features and Related XML Technologies

Namespaces

XML documents are everywhere and carry information or define data structures for information sharing. There is a possibility that similar names get used for tags in different XML documents which represent different elements. If the documents are read at the same time, it will be impossible to distinguish these elements. For example a <Table> in an xml document may provides data for a word processor which contains numbers or text. While an XML document which holds the information of the goods in furniture store have the same <Table> tag for a very different purpose. The idea of using namespaces was developed to solve these potential conflicts. A prefix can be defined to be added to the tags of the elements in that specific
namespace. This prefix is assigned to a namespace and all the child elements of such an element will be assigned to that namespace automatically. Following is an example of how a namespace is defined:

```xml
<?xml version="1.0"?>
<bk:Book xmlns:bk="http://www.civ.neu.edu/"
    <bk:Title> XML: Visual QuickStart Guide</bk:Title>
    <bk:Author> Kevin Howard Goldberg</bk:Author>
    <bk:Publisher> Peachpit Press</bk:Publisher>
</bk:Book>
```

In this example, `<Book>` and its children elements are assigned to the “http://www.civ.neu.edu/” namespace by `bk` tag. It is noticeable that namespace can be any phrase, and using web address doesn’t intend to make any connection to the website or any other web transaction. The idea of using internet addresses is accepted and widely used, because these addresses are unique for each company or developer and it will eliminate any possibility in using the same namespace in different documents. This way, two different elements from two different namespaces but the same names can be distinguished when used in the same document.

**XML Schemas**

As mentioned earlier, XML is an effective way for exchanging information in countless situations. Also it has the flexibility for defining customized elements and data structures. This capability has made XML an ideal choice for standard makers and experts in every field to provide standard data structures by which the documents can exchange their information. It is made possible by XML Schemas. A XML schema doesn’t contain any data, but defines a standard for the data and the way it should be presented in the XML documents which conform to that schema and allow
exchange of information. It is widely used to define the standard elements and data structures for different purposes. It has a set of its elements which are beyond the scope of this thesis.

The Associated General Contractors of America (AGC) has developed a standard XML schema, named agcXML, to be used as a standard data structure for exchanging transactional information in a construction project. Every XML data file that is supposed to be in conformance with agcXML, will be validated against it, and all the parties can be sure of the accuracy of their data exchange by conforming to agcXML in their XML data files. There are unlimited XML schemas which set rules for data structures for different business, scientific, and many more areas. The process in which an XML document is validated against a schema is called validation and if the document conforms to the schema, it is called a valid document.

The basic elements of an XML Schema

The following example shows the Schema definition for the XML example used throughout this chapter.

```xml
<?xml version="1.0"?>
<xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema>
  <xs: element name="Book">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Title" type="xs:string"/>
        <xs:element name="Author" type="xs:string"/>
        <xs:element name="Publisher" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs: element>
</xs:schema>
```
As in the above schema example, `<schema>` is the root element for every XML schema (Walmsley, 2001). The `<xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema>` states that the elements and data types are from http://www.w3.org/2001/XMLSchema namespace. Also all the elements from this namespace must have the xs: prefix. The phrase containing `<xsd:element>` defines an element and `<xs:complexType>` is used inside an element to define that it has children elements. These elements are called Complex Elements. In this example `<Book>` is a Complex Element. `<xs:sequence>` declares that elements Title, Author, and Publisher must be represented in the XML document in the same order as they appear in the schema.

Using this schema requires everyone to conform to its rule in providing information for their books.

To show the role of XML Schemas in AEC industry, In the following section some examples of well recognized and widely used XML schemas related to this industry are introduced.

### 3.3. SOME OF THE MOST RECOGNIZED XML SCHEMAS IN CONSTRUCTION INDUSTRY

“XML is one of the most widely-used formats for sharing structured information today: between programs, between people, between computers and people, both locally and across networks.” (Quin, 2010)

Using XML schemas can provide standard data structures for data exchanged in a specific business or trade. Different fields can be defined by which everyone is sure that their information will be shared completely, error free and rapidly. Following,
some of most widely used XML schemas which are developed and are used in construction industry are introduced.

\textit{ifcXML}

IFC is known as the backbone of BIM. It is widely used in 4D and BIM applications. While IFC has provided the required schema for saving and exchanging the information of a construction model, it lacks some benefits which could be achieved by XML. The most important benefit of using xml is its wide application in internet and different applications. It opens up new opportunities for IFC based systems to explore their domains into new applications and systems. It was the main reason for developing an XML Schema based on IFC, named ifcXML.

It is necessary to notice that IFC is not a counterpart for XML. XML and EXPRESS are both languages for defining data structures. EXPRESS is a standard data modeling language for product data. IFC is a standard model developed by EXPRESS and ifcXML represents the same model based on XML. There are countless instances based on each of these models which represent the related information in different construction projects. Figure 3.2 shows these relationships and categorizations.

The process in Figure 4.2 starts with a language like EXPRESS. This language is used to develop a standard data model, here IFC. This model is then used as a standard tool for end users to store the contents of its particular instances in a specific project in a standard way defined by the model. In Figure 4.2, “my door schedule” stores data about the schedule for a specific door, in the format and structure defined by IFC model.
Figure 3.2. The Process, Showed by Dash Lines, Starts with A Language for Model Definition, and Many Instances are Made from the Model (Liebich, 2002)

The instances from ifcXML are not supposed to replace the IFC ones. They can complement each other in meeting different needs of data exchange in a construction project. Some of the differences between a XML model (ifcXML) and EXPRESS model (IFC) which makes them both necessary are provided below.

- XML has a broader range of supporting utilities and is the standard data exchange model in the World Wide Web. This means that not only computers but cell phones and any other web-enabled devices can deal with XML data. Nearly all accredited general purpose applications (like Microsoft Office™) support XML format.
- An ifcXML file is significantly larger than the same file in EXPRESS SPF (STEP Physical File\(^2\)). It can be between 2 and 10 times larger (Nisbet & Liebich, 2007). Figure 3.3 shows a translation from a part of an IFC into ifcXML.

![Figure 3.3. Comparison of Data Definition in IFC and Ifcxml. (adapted from (Nisbet & Liebich, 2007))](image)

- The mapping to the XML schema definition necessarily loses some part of detailed information which are not supported due to the limitations of XML language comparing to EXPRESS (Nisbet & Liebich, 2007).

In general it is anticipated that the exchange of complete building models, particularly within a CAD environment will continue to be primarily based on SPF format. Therefore where large information sets (often including geometric models) are shared would remain the focus of the current SPF format. Other situations where partial models, reports, schedules or set of manufacturer information are exchanged, and the file size is not a major issue, or a messaging system of data exchange is being used, ifcXML will play the main role (Nisbet & Liebich, 2007).

\(^2\) The most widely used data exchange form of STEP.
Figure 3.4 shows the domain of IFC SPF format and ifcXML. The figure shows the benefits of using ifcXML in expanding the domain of AEC/FM into new business areas, because of the widely use of XML in these areas. These domains contain eCatalogs, GIS, eBusiness, and much more technologies based on XML and web. Technical guides and the free ifcXML schema is available online at www.buildingsmart.com.

Figure 3.4. The Anticipated Role of SPF and Ifcxml (Nisbet & Liebich, 2007)

aecXML

In an AEC project, there are basic information which defines the general aspects of the project. For example the name of the project, its location, the information about its owner, etc. aecXML is a XML schema which covers the general needs of Architectural, Engineering, and Construction (AEC) projects. The main purpose for developing aecXML was to improve the communication among different parties involved in a construction project, by developing schemas for the exchange of AEC-

aecXML can be considered as a foundational tool which was developed to cover the general information exchange in a construction project. For example, agcXML is specifically designed to cover the information which is exchanged in AGC documents and widely uses aecXML elements and namespace. An XML schema in itself may refer to other schemas.

**agcXML**

A construction project produces a huge amount of data which is circulating among different parties involved in the project. Large portions of this information are reproduced and typed into different software and/or documents which mean a lot of waste in time and efficiency and results in many inconsistencies. IFC classes support the modeling and geometrical data in a project. Beyond the model, there are still huge amounts of transactional and business information which are produced and exchanged in a construction project and are not usually covered sufficiently by IFC classes and the related software. agcXML is a set of XML schemas which facilitates the exchange of the afore-said data. It facilitates the exchange of information commonly included in the following document types:

- Request for Information (RFI)
- Request for Pricing/Proposals
- Owner/Contractor Agreements
- Schedule of Values
- Change Order
The agcXML Project is an industry-wide initiative funded and led by Associated General Contractors of America (AGC) and managed, under a contract with AGC, by the National Institute of Building Sciences, within the framework of the IAI’s³ aecXML Domain.⁴ It uses aecXML for defining all its abstract elements that make general resource of AEC specific elements, like the specification of a project or the id for every element which might be used to refer an object by another object, and adopts many elements from ifcXML (BuildingSMART alliance, 2008)

**GreenBuilding XML (gbXML)**

gbXML was designed to improve the information exchange between CAD and design software and engineering analysis programs, mainly in energy simulation and analysis domain. It is officially introduced as:

“The Green Building XML (gbXML) open schema helps facilitate the transfer of building properties stored in 3D building information models (BIM) to engineering

³ International Alliance for Interoperability, www.buildingsmart.com

⁴ Technology Page at: [http://www.age.org/cs/industry_topics/technology/age_xml](http://www.age.org/cs/industry_topics/technology/age_xml)
analysis tools. Today, gbXML has the industry support of leading 3D BIM vendors such as Autodesk, Bentley, and Graphisoft" (gbXML welcome Page, 2010).

It was adopted by most CAD vendors in their products and was added to some of their leading products. Its main application is in integrating energy analysis software with 3D CAD application which makes data exchange automatic, leading into the possibility of deciding the optimum design considering energy efficiencies, between different models in the preliminary phases of design. It will reduce the time and cost of analyzing different models (Cover, Green Building XML (gbXML), 2004).

IFC and gbXML are both mature models and are supported widely. Obviously, the differences between XML and EXPRESS languages apply here, too. IFC uses the relational approach of EXPRESS and defines complex and complete structures for defining all the details, while gbXML provides an easier approach while not covering all the details and relationships. As an example, IFC can support any kind of shape for building elements, while gbXML only covers rectangles, which is usually sufficient for energy simulation (B. Dong, K. P. Lam, Y. C. Huang, Dobbs, & M2, 2007).

3.4. THE DOCUMENT OBJECT MODEL (DOM)

The Document Object Model is an object oriented representation of the documents that can be used as an interface by programming or scripting languages to update the content, structure and style of the documents (Le Hegaret, Whitmer, & Wood, 2005). XML DOM is the part of DOM that facilitates the access and manipulation of XML documents.

XML DOM defines “nodes” as the building block of every XML document. For example “document node” represents the XML document itself. Element node
represents an element in an XML document, and “text node” is the textual content in an element or attribute. The most important node types are Element, Attribute, Text, Comment, and Document (w3schools, 2010). The nodes can be distinguished by 3 important properties of Node Type, Node Name, and Node Value. DOM defines the XML document as a tree structure where the nodes can be accessed in 3 different ways:

- Using getElementsByTagName() method to get the elements with the defined Tag name
- Looping through the nodes tree
- Navigating the node tree, using the node relationships. Relationships between the nodes are defined by the following properties to the nodes:
  - parentNode
  - childNodes
  - firstChild
  - lastChild
  - nextSibling
  - previousSibling

Different packages have been developed for using XML DOM in programming languages such as Java. Some of them are open source, free and are widely used. Figure 3.5 shows the tree model of the XML representing Book data.
3.5. WEB SERVICES DEFINITION LANGUAGE (WSDL)

XML has opened up unlimited opportunities for facilitating data exchange. There are situations where an application may need some common data which can be found in the standards or in the public sector, or in some cases a group of applications may need a source of information on the web. Web Services Definition Language is an answer to this need. For example, there are web services available which can give the current rates for currencies upon request, or give the zip code of a location, in an XML format to be directly used in an application or web page.

WSDL is a XML language for defining and accessing web services. It defines the location of the service and the operations it provides. A WSDL document contains these 4 major elements (w3schools, 2010):

- `<types>` defines the data types used by a web service
- `<message>` defines the message used by a web service
- `<portType>` defines the services provided by a web service
- `<binding>` defines the communication protocols used by the web service.
There are numerous situations that can be improved using web services. An example is Universal Description, Discovery, and Integration (UDDI). UDDI is a community of over 220 companies which provides numerous web services for members. UDDI provides general purpose services, while there are some BIM related web services available. One of these cases is ONUMA Web Feature Service\textsuperscript{TM5}. This service works based on open standards from Open Geospatial Consortium (OGC) and can be used for providing the geographic information of a specific area and the features inside that area, like buildings, that are available in ONUMA system. It can list BIM, IFC, and City GML data that is available within the boundaries of a geographic area.

3.6 SUMMARY

This chapter provides a basic understanding about XML and its related technologies to the reader. It first described the importance of XML in data exchange and especially through internet. Then it gave a brief introduction to the syntax of XML, and introduced Namespaces and XML schemas as two important XML technologies. To increase awareness about the role of XML in AEC industry, some of the most recognized XML schemas for this industry were introduced. The coding in the system developed for the thesis uses DOM, so a part of this chapter was devoted to introduce this technology. WSDL was also introduced as an advanced tool which can be used in Smart forms.

\textsuperscript{5} http://www.onuma.com/products/WebFeatureServices.php
CHAPTER 4. DATABASE DESIGN PRINCIPLES

4.1. PRELIMINARY STEPS FOR DESIGNING A DATABASE

The main objective of this thesis is centered on developing an IT solution for the construction industry to improve the process of gathering data by fillable forms. This solution uses different tools and applications such as pdf forms, database definition and data management, some codes for manipulating data and performing different commands in a system and many more IT relevant situations. A database was developed for storing the data gathered by the forms.

Just like any other field, developing IT and software solutions requires preliminary studies before starting to develop the product. There is sometimes a tendency to spend insufficient time researching the problem and to start developing the code, database or other software solutions. Not taking enough time for preliminary studies and business requirement analysis has caused many software projects to be late, over budget, or even fail. McConnel introduces a list of 36 classic mistakes, in four groups of people-related, product-related, technology-related, and process-related mistakes in a software development project that leads into unexpected results and a poor product (McConnell, 1996).

Concerning the importance of business analysis and preliminary studies in clearly understanding the business requirements in a software project, the first part of this chapter introduces two methods as the methods for developing these studies. One
method is what is defined and used by National Building Information Modeling Standard (NBIMS) which is mainly used for developing IFC solutions, but the concepts are so general that seem appropriate for any other software project related to construction industry. The benefits of using this method and developing the related documents based on the guidelines and the documents introduced by this standard is that the same standard can be used for all related software solutions in the construction industry.

The second method, business analysis, is used in software industry for distinguishing the problem and the business needs. It is beneficial to understand that all these methods use the very same procedures and concepts. They may have only minor differences in the details required, and so on. They are all based on performing research about the business in different ways like interviews, questionnaire, reading the resources and other ways. Each method then contains a proposed for documenting these findings, and the ways and practices to develop models based on these distinguished requirements. In the following these two methods are briefly explained.

4.2. NATIONAL BUILDING INFORMATION MODELING (NBIM) DEVELOPMENT PROCESS

The National Building Information Modeling Standard (NBIMS) Version 1, Part 1 was published in 2007. This document includes contributions by more than 30 subject-matter experts in the capital facilities industry and is developed under the supervision of buildingSMART alliance which is a council of National Institute of Building Sciences. This document provides an overview, principles, and methodologies used for developing BIM solutions. The standard defines means and methods which should be undertaken for developing BIM standards for different
aspects of businesses which use building information. Figure 4.1 shows the steps which should be undertaken for Exchange Standard development and use process (buildingSMART alliance, 2007). Following these steps and guidelines results in two documents. The first one is Information Delivery Manual (IDM) and the second document is Model View Definition (MVD).

**Figure 4.1. Overview of Exchange Standard Development and Use Process as Defined In NBIM Standard Version 1, Part 1**

IDM is the exchange definition written in non-technical format for use by end-users. It describes the business process, stakeholders, exchange points, information requirements and business rules. (buildingSMART alliance, 2007). As an example of an available IDM document form buildingSMART alliance, the IDM for Quantity Take Off (QTO) contains these parts (bSa & OGC, 2009):

- Process maps and specification of processes
- Specification of data objects
- Exchange Requirements (ER)
More information about IDM can be found in the IDM development guide (buildingSMART alliance). MVD is developed from IDM and is the technical exchange definition for use by software developers.

4.3. Business Analysis for Developing Database Solutions

There are different methods for conducting the business analysis for establishing the requirements which should be addressed by a software solution. It may include studying the related resources for getting to know that trade, conducting formal and informal interviews, questionnaires, surveys, and other methods.

An executive summary of the business characteristics and needs is then developed based on the findings from preliminary studies. An Executive Summary is a short document which introduces the business and the requirements which are needed to be addressed by developing the database. The next level may be developing the Entity-Relationship (E-R) diagrams, which is introduced later in this chapter. A data dictionary is another document which might be required. This document gives information about the tables and fields used in the database, their data types, their possible key function, possible default values and other information which might be required.

4.4. Database or Spreadsheet?

Designing a system for storing and managing data is an important step. Spreadsheets, such as Microsoft Excel, are common tools for storing and managing data in construction projects. The common database system used is Microsoft Access. There are several reasons that an Excel spreadsheet may be preferred to an Access database.
Most general users are more comfortable and familiar with Excel and having all data accessible in one place, are the two most important reasons for preferring Excel spreadsheet. There are some guiding points that can help in deciding between a Database Management System (DBMS) and a spreadsheet solution. The following guidelines can be helpful in deciding whether to use Excel or Access for storing and managing data (Nelson, 2010):

- When the data are relational the use of a DBMS is usually preferable. This means that the relationship between pieces of data requires more than one table. This happens when data of more than one entity is to be captured. For example, keeping data for construction material needs only one table. Obviously it is not sufficient for a construction project because at least another entity, the related merchants, is required to manage the works.

- When there are large volumes of data, such as long strings for each cell in the spreadsheet, managing them in a spreadsheet is a difficult task.

- If there is a chance that more tables will be added in the future, especially in a relational format, database must be considered.

- If data mostly contain large text strings, which makes it difficult to see the whole data in different cells at one time, it is advised to use a database. In a spreadsheet, data are always exposed and this makes reading, navigating and managing data difficult. Large strings of data, usually a complete sentence or more, will make the spreadsheet messy and difficult to navigate through data which might not be needed in most of the time.
• If there is a need for running complex queries on data, which contains multiple conditions for searching and gathering data from different sources, using a database is much more efficient.

• If the system relies on external data from other database systems a database must be used to be able to define the connections.

• When a group of people are working with data, a DBMS can provide different levels of access to data for different groups of users.

Any of the following conditions may be sufficient to consider a spreadsheet for managing data:

• When the size of data is known to be small enough to be manageable in a spreadsheet. Means that the data in the related cells are short enough that can be easily read together.

• When data are flat or non-relational and can be captured in one table.

• When some primary calculations or statistical comparisons are being done on the data. As explained the data are exposed in a spreadsheet, also the spread sheets contain different functions to the many kinds of calculations on data.

Considering the above mentioned conditions for the case of this thesis, the database system was selected over the spreadsheet for storing the data gathered by the forms. In the following the concepts of designing the databases are described.

4.5. DATABASE CONCEPTS

Having the right information at the appropriate time for the appropriate person or organization is one of the most important issues in an efficient business. The
availability of data in the shortest time when needed and the integrity of data are important factors in business superiority. Construction industry uses a large amount of data over the lifecycle of a project. It starts from information in conceptual design step, to detailed design parameters and legal issues contained in a contract and continues in the construction process to provide information about the work improvements and different site information. The flow of data continues to exist and develop in the service and maintenance period of a facility. The information produced during the project life needs ample attention to be managed effectively and to be accessible when needed.

4.5.1. Lists and Related Tables

The easiest way for recording the data for a situation is to add them to a list. New data can be placed in new rows. Figure 4.2 shows a list that keeps the data of materials used for construction work in a site. The list is developed in accordance with OmniClass definitions and numbering system and stores basic data about material.

Figure 4.2. List of Materials with Their Omniclass Numbering System

“OmniClass is a multi-table faceted classification system designed for use by the capital facilities industry to aid sorting and retrieval of information and establishing classifications for and relationships between objects in a building information model” (buildingSMART alliance, 2007). Changing the data in the list can be easily done.
One may change the values, add new material or remove a material from the list. It can be sorted alphabetically or by the numbers. Facing no difficulties in changing data shows that spreadsheet is enough for performing all these tasks and other data manipulation tasks. To be effective in a realistic work situation, the contractor may need to add more information such as some information about the merchants who provide the materials. Figure 4.3 shows the revised list containing the new information.

Figure 4.3. List of Materials and the Providers

The improved list still works well for sorting and data analysis. But the data usually change in amount and value. In these cases using a list for storing data will face modification problems. The modification problems may occur when more than one entity is included in one list, like the one in Figure 4.3 which represents data about material and their providers in one list. Three types of modification problems may happen when more than one entity is included in one list: (1) deleting a row of data, (2) changing data in a row, and (3) adding new data to the list. Figure 4.4 shows the situation that may happen for our list. In this example ARDEX Americas is not considered as the cement provider in a new project and should be eliminated from the list. As Figure 4.4 shows, erasing row 4 in the table will also erase data for Cements which is not desirable. An update problem happens when, for example, Jewson Builders, announces a change in their contact email address. Changing the email
address in row 6 will cause inconsistent data problem with the email address available in row 7.

Another problem happens when HIGHBEAM Business has given an offer to provide a range of different materials. Although the contractor has not reviewed and decided on the offer, he may want to keep the contact information. Inserting contact data of the new merchant in row 8, Figure 4.4, will cause a row of incomplete values, called null values. Null values are always problematic and should to be avoided.

In a real situation there are numerous materials provided from different merchants. One merchant might provide several different materials, and a product might be bought from different providers. The modification problems will be extremely complicated in such situations and it was the driving force for developing database concepts in 1960s. From different methods considered for solving the mentioned problems in lists, relational model emerged as the leading solution and today almost every commercial database is based on the relational model (Kroenke & Auer, 2010). In the following the concepts of relational models are explained.

4.5.2. Relational Models

A relational model is made of different tables and the relationships between them. It is developed based on the simple and obvious idea of keeping the data related to each
entity in one table and provides techniques, guides, and development strategies to do this task on different complex business situations. To illustrate the benefits of using relational model, the table in Figure 4.4 has gone through the process of “separating the entities”. It contains information about two different entities, materials and merchants. So it can be decomposed into two tables, as shown in Figure 4.5. It can be found out by testing, that none of the mentioned modification problems associated with lists and related tables will happen for these tables. A link between the two tables is also required for finding the provider of a specific material or tracking the materials provided by a specific merchant. For linking the two tables, the material name is added to the merchant table. As it is shown in Figure 4.5, data are easily trackable between two tables. In general, to evaluate a design, three basic modifications can be used for doing testing the tables. One must be able to insert, delete, and update in these tables with no modification problem, as explained in the previous section. It is a good practice to consider the same name for the linked fields in both tables, but not necessary.

Figure 4.5. The Relational Model of MERCHANT and MATERIAL Tables
4.5.3. A Database System

The issues explained in the previous section contain the basic idea behind developing relational database systems and methods. A database system is composed of four components: Users, the database application, the database management system (DBMS), and the database (Kroenke & Auer, 2010). The database is a set of self-explaining related tables, and DBMS is the application or set of applications which are used to create, process, and administer the database, like Microsoft Access™ which was used for developing the database used in this thesis. It should be mentioned that in database design principles, the lists that store data about different instances of one entity are called “table”s. For example Figure 4.5 shows two tables called Merchant and MATERIAL. A table may be shown in text or graphic format. For example MATERIAL and MERCHANT tables, Figure 4.5, are shown as follow:

| MATERIAL (Number, Name, Type, Provider) |
| MERCHANT (Merchant, Email) |

The features expected from a DBMS are as follows (Kroenke & Auer, 2010):

- Create database
- Create tables
- Create supporting structures like indexes
- Read database data
- Modify database data
- Maintain database structures
• Enforce rules
• Control concurrency
• Provide security
• Perform backup and recovery

In the follow, the steps in designing a relational model are described.

4.5.4. The Basic Steps in Designing a Relational Model

In the previous sections the problems of using lists for storing data were explained. Based on these facts, the relational models were introduced as an appropriate solution for solving these problems. In this part, the steps for developing a relational model for storing a set of data are discussed.

As explained and shown in Figure 4.5, the table of Figure 4.4 is decomposed into new tables until each table represents one entity. A part of decomposed table must remain in the root table to maintain the relationship between the two tables. It is named key field. A key is one or more columns in a table which identifies a row. If a non unique key from the referenced table is added to the referencing table, there is no way to distinguish which row should be addressed. So, a unique key from the referenced table is added to the referencing table. This key is known as a foreign key in the referencing table.

In this example, the name of the merchant is considered as a key to be planted in the MATERIAL table. It is based on the assumption that the names of the material providers are unique, so each key references one and only one row in MERCHANT table. This rule can be defined in the following phrase:
Such a rule is called a referential integrity constraint. Whenever there is a foreign key, here Provider in MATERIAL table, an associated referential integrity constraint must exist, which as mentioned before, is applied by the DBMS. The values in a foreign key cannot be changed in the referencing table, but can be changed in the referenced table and the DBMS changes the corresponding value in the foreign key. Finding the appropriate foreign key and setting the referential integrity constraint needs enough knowledge about that business and the relationships among data.

4.5.5. Functional Dependencies and Normalization

The subjects explained in the previous section were the basics of database design. The key point in decomposing data into new tables is to find functional dependencies. For example, knowing the OmniClass number of a material is enough to find its name and its type. This relationship can be written as follow:

$$\text{Material number } \rightarrow \text{(Name, type)}$$

Also having the name of a company gives its contact number, address, email and so on.

$$\text{CompanyName } \rightarrow \text{(Address, Phone no., Email)}$$

In this situation it is said that Address of a company is functionally dependent on its Name. The same can be said about Phone number and Email address. Also, CompanyName determines Address, Phone Number, and Email address and is called a determinant. Considering the definitions and examples of relations and functional dependencies, it can be defined that relations are to store instances of functional dependencies. The process of breaking a relation into more relations that each
represents one functional dependency is called *normalization*. The improvement from one table in Figure 4.4 to two tables in Figure 4.5 is a simple example of normalization. Finding the dependencies in datasets needs enough knowledge about that business and its requirements which can be obtained through the processes mentioned in the beginning parts of this chapter.

4.5.6. Primary Keys and Candidate Keys

The role of keys in establishing the connection between tables was explained before. There are two types of keys in the referenced table, Candidate Keys and Primary Keys. Candidate Keys are considered as "one or more attributes which are unique in each row and functionally determine all other attributes of the relation" (Kroenke & Auer, 2010). There might be more than one Candidate Key in a table and one of the Candidate Keys is considered as the Primary Key. A Primary key is to be short, numeric, constant and always has a value. Consider this example of a relation called EMPLOYEE:

EMPLOYEE (EmployeeNumber, FirstName, LastName, Department, Email, Phone)

The following functional dependencies can be found in this relation:

EmployeeNumber \(\rightarrow\) (FirstName, LastName, Department, Email, Phone)

Email \(\rightarrow\) (EmployeeNumber, FirstName, LastName, Department, Phone)

(FirstName, LastName, Department) \(\rightarrow\) (EmployeeNumber, Email, Phone)

EmployeeNumber, Email and the set of (FirstName, LastName, Department) are Candidate Keys of this relation. It means that knowing their value, the unique values of other attributes can be found. From these candidate keys, one which is the most
important and meaningful for the database users, is chosen as the *Primary key* and is used as the foreign key in the tables which are supposed to refer to EMPLOYEE table. The concept of functional dependencies is used to Normalize the data into different tables, which is the subject of the next part.

4.5.7. Normalization

Understanding the concept of functional dependency, *normalization* can be defined as breaking a table with more than one functional dependency into tables which have one and only one functional dependency.

For normalizing the relations the following process can be considered (Kroenke & Auer, 2010):

1. Distinguish all candidate keys of the relation.

2. Determine all the functional dependencies in the relation and their determinant. If any determinant is not a candidate key the relation is not well formed and must be broken down. In this case:

   a. Extract the attributes of the functional dependency from the relation and put them in a new relation

   b. Consider the determinant of the functional dependency as the Primary Key for the new table

   c. Put a copy of the determinant as the Foreign Key in the original relation

   d. Create a referential integrity constraint between the original relation and the new relation
3. Repeat step 2 until every determinant of every relation is a candidate key.

Understanding the working process in a business is a key point in distinguishing the functional dependency. Wikipedia defines normalization as follows:

"a systematic way of ensuring that a database structure is suitable for general-purpose querying and free of certain undesirable characteristics- insertion, update, and deletion anomalies- that could lead to a loss of data integrity."

There are different levels for data normalization, called normal forms. The higher the level of normalization, the less will be the probability of anomalies in the database. There are eight levels of normalization available, which are called Normal Forms and each higher form adds a criterion to its previous one. The process explained above is used to obtain the First normal form for a database (1NF). In this project the database is 1NF which is sufficient for most simple databases.

4.5.8. Data Modeling and the Entity-Relationship Model

Normalization is the usual method for dividing data into related tables. Before developing the real database, the tables and their relations are modeled in diagrams. It is similar to the role of flowcharts in programming. The most popular data modeling approach is Entity-Relationship (E-R) model. An entity is something that the user wants to track and gather information about. For example MATERIAL is an entity which its Name, OmniClass Number, and type are required for our case. These three pieces of data are called attributes of MATERIAL entity. When defining these entities for a database, the entity is called table and attributes are called columns. In general, the steps in transforming a data model into a database design contain creating a table for each entity and distinguishing the primary key for that table. The
ideal primary key is short, numeric and constant. There are instances when finding a primary key which can have the entire mentioned characteristic is difficult. For example, in the following EMPLOYEE table; there is a chance that there are two or more employees with the same name.

EMPLOYEE (FirstName, LastName, Department, Phone)

In this situation the following relation can be considered

(FirstName, LastName, Department) ➞ (Phone)

That defines (FirstName, LastName, Department) as the primary key for this table. It is far from the ideal primary key which was defined earlier. In these situations using a numbering system as the primary key might be the solution. For example each employee might be given a unique Employee Number which is short (one field), numeric, and constant. Another solution is using Surrogate Keys as the primary key in the tables. Surrogate Keys are numeric and unique numbers created automatically by DBMS and usually invisible to the user. They have no meaning related to the entity represented by the database and their use is not advisable.

In this example, the EMPLOYEE table can be revised by adding Employee No. as the primary key which determines all other attributes in the entity:

Employee No. ➞ (FirstName, LastName, Department, Phone)

4.5.9. Specifying Properties for Each Column

Data provided in a table may have different properties. Setting these properties makes data manageable and can provide control over data. When designing the database, the following properties are defined for each table (Kroenke & Auer, 2010).

- Data type
- Null status: it defines if an attribute is required to have a value when a new row of data is created in the table. If Null status is Yes, a new row can be added to that table without having a primary value for that attribute.

- Default Value: some attributes might be considered for having a default value. For example in a database related to a specific project, a default value might be set for the ProjectName column in any table containing this column.

- Specify Data Constraint (if applicable): for example a ZipCode might be checked to be a 5 digit number and other values not be accepted.

Also each table goes through the normalization process to make sure all tables are normal. As a standard rule, in text representation of a table, primary key(s) are underlined and foreign key is in Italic format.

### 4.5.10. Cardinality

A database works as it is expected when the relationships between the tables are defined. One part of this relationship is to define and put the *foreign keys* in the referrer tables to connect tables in a meaningful way. The next step is to define the number of rows in a table which can be connected to one row of the other table. This relationship is called Cardinality.

Figure 4.6 represents the Crow’s foot symbols of the relational model and Figure 4.7 is an example of this kind of E-R diagram that contains MATERIAL and MERCHANT tables.
### Figure 4.6. Crow's Foot Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Symbol]</td>
<td>One-Mandatory</td>
</tr>
<tr>
<td>![Symbol]</td>
<td>Many-Mandatory</td>
</tr>
<tr>
<td>![Symbol]</td>
<td>One-Optional</td>
</tr>
<tr>
<td>![Symbol]</td>
<td>Many-Optional</td>
</tr>
</tbody>
</table>

### Figure 4.7. The Relationship Between MATERIAL and MERCHANT

There are symbols at both ends of the connecting line. The symbol closest to the table defines the maximum cardinality and the other one represents the minimum cardinality of an entity in a relationship. *Cardinality* is an important part of the relationships in a database and defines the number of rows from a table which is supposed to be addressed by one row of another table. In Figure 4.7 MERCHANT has the minimum and maximum cardinality of one. It means that for each new MATERIAL there must be one and only one MERCHANT defined and related to that material. This is also called One-Mandatory cardinality. On the other side, the minimum and maximum cardinality for a MATERIAL entity are zero and many respectively. It means that a merchant may not be attached to any material, and when attached it can provide many materials for the project.
Three types of relationships can be defined between two tables (Kroenke & Auer, 2010):

- The one-to-one (1:1) relationship
- The one-to-many (1:N) relationship
- The many-to-many (N:M) relationship

Using zero instead of one in any of these relationships defines if that row is mandatory or not. Four different symbols are defined in the crow’s foot diagramming to show the relationship and the cardinality of the tables. These symbols are shown in Figure 4.6.

The dashed line connecting the two tables in Figure 4.7 shows that both tables have strong entities. It means that the existence of each table is not dependent on the other one. There are some entities which their existence in a database is dependent on another entity and are called weak entities. As an example of Weak Entity, consider APARTMENT and BUILDING entities. An APARTMENT cannot exist without a BUILDING, therefore it has a weak entity. The weak entity relationship between two tables is shown by using a solid line for connecting them, in their Crow’s Foot diagram.

In Figure 4.7 (FK) means Foreign Key, and the primary keys are shown in the top part of each table, separated from other attributes by a line. The connector of the tables shows their cardinality and type of the relationship. There are other ways for representing the relational model which are not in the scope of this thesis.
4.6. SUMMARY

The steps in designing a database were introduced in this chapter. Like any other project in every business and industry it starts with doing the research on the business and finding out the business requirements and the processes. The result of this research might be an Executive summary or an IDE as defined by NBMIS version 1. This chapter gave a brief introduction to the principles of database design. The objective was to familiarize the reader with principles like Normalization, primary keys and foreign keys, referential integrity, and cardinality. Knowing these principles gives enough tools to start designing simple databases and understanding a designed database such as what was developed for this thesis.
CHAPTER 5. THE TECHNOLOGY OF DYNAMIC PDF FORMS

5.1. WHAT ARE DYNAMIC FORMS

The pdf forms which were used as a testbed for developing the solution for a Smart Form system are dynamic forms which their functionality is described in Chapter 2. This section describes the dynamic forms.

Traditional forms were laid on papers, requiring the user to write data in each required field. Dynamic fillable forms address this situation by providing changeable forms which can change in size and functionality in different situations. A static form contains a number of field objects that are fixed in length and all different situations must be fulfilled by these forms and the required fields be provided, but those situations may need different parts of the form to be filled and some parts left blank. One solution for avoiding large blank fields might be to design various versions of the same form to cover different situations. Tax return forms are good examples which are widely used and come in different versions to cover different needs. Form 1040 and its simplified version, 1040 EZ is an example of this approach. Form 1040 covers all possible taxable situations, while its simplified form, 1040EZ is good for the simplest situation. But these 2 forms just cover two ends of a spectrum and there are many cases that have to be addressed. Sometimes, requiring just a single additional
data to Form 1040EZ, will necessitate that the tax payer switch to the complicated 1040 form and leave most of the fields blank.

The shortcomings of a static form can be summarized as:

- Static forms can cause considerable paper waste
- A large number of unrelated fields in the form may distract the user, resulting in waste of time and inadvertent mistakes
- Reviewing the static form with many blank fields is more time consuming and less efficient

A dynamic form addresses the above issues by providing a changeable interface which can be adjusted to fit different amount of data and needs of the user. The goal is to have the least possible or no field left blank in the form. It results in increased efficiency and reduced cost.

Pdf forms are widely used and can provide features of dynamic forms. But these forms are typically not dynamic forms. The dynamic features are added to a pdf form by using Adobe Livecycle Designer™. Livecycle Designer™ is a product from Adobe™ for designing dynamic and data-connected pdf forms. The forms built by this application have the advantage of XML technology by providing XML models of the forms and the ability of connecting XML data files and schemas to the forms. Dynamic instances with forms occur under two basic conditions (T. Padova, A. Okamoto, 2009):

- When data are imported into the form the fields can be expanded to create the required field and page numbers to include the data. The structure of form is not changed in this situation.
• The second condition occurs when the form changes in its form and structure based on different situations which the form user selects, usually by pushing buttons or other Graphical User Interface (GUI) controls.

5.2. THE BUILDING BLOCKS OF DYNAMIC FORMS

Dynamic Pdf forms are made dynamic by using subforms in Livecycle Designer. Subforms are used for creating flowing dynamic layouts, controlling the tab sequence, and data binding. Subforms allow the use of dynamic forms that grow and shrink in Livecycle Designer (referred to as Designer from this point on). The dynamic tables are also considered as specialized subforms which provide the ability to add or delete rows in the table. This change in size may happen by the flow of data into the dynamic table or by providing buttons or other interface tools to let the user control the number of rows at run time. For both situations the table must be defined with varying body rows in Table Assistant of Designer. For the former situation one or more data connections will be defined and in the latter situation some coding is required to do the necessary functions. The tools used to make the forms dynamic are explained in the following section.

5.3. MAKING THE FORMS DYNAMIC

Different kinds of dynamic features have been added to the forms in the electronic version of ACRP Report 21 forms and templates. Below, these features are explained.
5.3.1. Subforms

As mentioned earlier, a form can be made dynamic by using subforms. In Part 2 - Analytical Approach, subforms have been used widely for this purpose and also for keeping the related items in the same groups.

As shown in Figure 5.1 the form is composed of three main parts; (1) first page, (2) pagGoNoGo and (3) Tier1. At the runtime Tier1 is hidden, so the form contains only 2 initial pages. After selecting the applicable delivery methods in Go/No-Go table, Figure 2.1, and pressing the Finalize Go/No-Go button, a line of Javascript code makes the Tier1 subform visible. Depending on the selected delivery methods the new size of the form may vary from 27 to 44 pages. The mechanism is explained below.

Each delivery method has an advantages/disadvantages table for every issue. Figure 5.2 shows the structure of Issue 1 subform. This structure is the same for all 19 issue subforms. It contains the titles, which is Issue 1: Project Size/Complexity for Issue 1, then four tables containing the issues for each delivery method. The tables are DBB, CMR, DB, and DB2. The txtSummary Text Field contains Summary word, and the five tables following that, all look to be in one row of the Summary table and are used to evaluate different methods for that issue. Figure 2.3 is an example of Issue 1 subform. In Figure 2.3, Design-Bid-Build and CM@Risk are considered for comparison, so tables DB, DB2, tblDBSumm, and tblDB2Summ in Issue 1 subform are hidden. Based on the selection in Go/No-Go table, the tables for delivery methods that are not applicable to the analysis will be hidden from view. The “presence” property of tables DB and DB2 are set to hidden for all issues in this analysis.
Figure 5.1. The Hierarchy of The Form Part 1-Analytical Approach

Figure 5.2. The Components of Issue1 Subform
5.3.2. Dynamic Tables

Dynamic tables can be designed for many uses and different models. A table that dynamically creates rows and/or columns on a single page is one example of a dynamic table while for another model one can create tables that flow to additional pages along with any headers or footers assigned to the table (T. Padova, A. Okamoto, 2009). In Designer, tables are a specialized subform which makes them ideal for repeating sections of a form. Two types of dynamic tables were developed in the forms used in this project.

5.3.3. Table with Variable Number of Rows

In many situations the number of data sets used in a table is unclear at design time. A dynamic table with adjustable number of rows can provide the required efficiency and sufficient space for containing the required data.

Form 1 uses a table of this kind for major schedule milestones (Figure 5.3) of the project. By pressing “Add Milestone” and “Delete Milestone” buttons, user can adjust the number of rows to the number of milestones. A minimum of one milestone is always required. For nearly all projects, at least two milestones, start date and completion date, are required. The table is created by using “Insert Table” option in Designer and choosing the variant “Body Rows” option for the table in table assistant.

<table>
<thead>
<tr>
<th>Major Schedule Milestones</th>
<th>Add Milestone</th>
<th>MileStone</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete Milestone</td>
<td>Notice to Proceed (NTP)</td>
<td>Mar 20, 2010</td>
<td></td>
</tr>
<tr>
<td>Delete Milestone</td>
<td>Start of construction</td>
<td>Apr 17, 2010</td>
<td></td>
</tr>
<tr>
<td>Delete Milestone</td>
<td>Substantial Completion</td>
<td>Nov 18, 2010</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.3. The Dynamic Table for Adding Milestones to the Part1-Specifications Form
Pushing “Add Milestone” button triggers the *Click* event of the button. Then the following JavaScript code adds a new row to the table

```
_row1.addInstance(true);
```

In the above code, Row1 is the first fillable row of the form and addInstance creates a new instance of Row1. Instance Manager in Designer contains *Add*, *Remove*, *Set*, and *Move* methods which are used to control the instances of subforms or table rows. moveInstance method of this group is used for deleting every single row of the table when the appropriate button is clicked.

### 5.3.4. Table with Variable Number of Columns

In Part2-Analytical Approach, the summary tables are actually a set of five tables which are put next to each other and create the view of one table. These tables are *tblSummary*, *tblDBBSumm*, *tblCMRSumm*, *tblDBSumm*, *tblDB2Summ* (Figure 5.). Depending on the selection in the Go/No-Go table, some tables are set to be hidden because they represent non-applicable PDMs. For example, in Figure 2.3 in Chapter 2, DB methods are not applicable, so *tblDBSumm* and *tblDB2Summ* tables are set to hidden and omitted from the Summary.

### 5.4. SUMMARY

This chapter described the methods in designing dynamic pdf forms and used examples from the Smart Form system to make the points clear. The intent was to provide information to the reader on what happens behind the scene of a dynamic pdf form and the ACRP Report 21 forms. The available tools and methods for making a
dynamic pdf form by LiveCycle Designer™ were explained. These tools are subforms and dynamic tables.
CHAPTER 6. THE ACRP REPORT 21 SMART FORMS PROJECT

6.1. INTRODUCTION

Data for major construction projects are typically gathered either manually or electronically. Forms are one of the most common methods for sharing and exchanging information in construction projects. These are the first and most lasting information carriers. A project is tracked before any design is undertaken and long after geometrical models (Drawings and 3D models) have fulfilled the expected duties.

The traditional ways of using fillable forms contain the following inefficiencies that this thesis aims to solve by developing a framework containing different software tools:

- An AEC project consumes a large amount of paper for fillable forms, that means consuming natural resources. Using electronic forms, as the fundamental part of Smart Form system, considerably reduces this waste.

- The traditional way of collecting data in paper forms requires personnel to enter data into different systems. Even in case of electronic forms, someone has to enter the data. A Smart Form system, subject of this thesis, can eliminate the use of personnel in this process by providing a method for automatic data transfer from the form to the data repository.
In the traditional method of using forms the same data may be entered by different individuals in different software systems. This approach increases the chance of having different values for a single data in different places, and modifying them can be a very difficult task. Construction Smart forms relies on XML technology as a widely used standard for data exchange. It means that data gathered by a Smart Form is directly accessible by almost all the available software, and can be put in a single repository which makes it easy to manipulate, and reduces the likelihood of error.

Considering the mentioned goals for a Smart Form system, a software system containing different software tools was developed as a template for Smart Forms.

Many of the ideas about Construction Smart Forms were applied to the Part1-Specification pdf form from ACRP Report 21 forms. The forms were then further enhanced by designing a system to manage the data collected. As part of this system a database was designed as the storage place for data gathered by the forms. This chapter introduces different possible solutions which can be used for this purpose and explains the solutions used for this project. Figure 6.1 shows the process used for this thesis, which starts from exporting data from the form to the xml document and ends with importing the revised xml file into the database.
6.2. An Explanation of the Smart Form Template Developed for This Thesis

This chapter describes the specific Smart Form solution which was developed by the author. Figure 6.1 shows an overview of the developed system. The process starts with a dynamic fillable pdf form. It can be an electronic version of any traditional fillable form which is used in an AEC project. The reason for choosing the pdf forms is that they are widely used all over the world and is considered as the most accepted standard format for electronic portable documents. Also the recent pdf files support XML technology which is a fundamental part of a Smart Form. The important thing is to define the elements of the form in an appropriate way, so the resulting XML file,
will need minimal changes to be imported to the database. Section 6.4.3 gives detailed information about this issue.

After entering data in the form it can be exported into an XML document. It is easily done by an “export” option available in the Acrobat Reader™ software. The resulted XML document contains all data provided by the form recipient but is not ready to be imported into the database. This author has developed three modules in Java programming language which transform the initial XML document into a document which can be imported into the database. Java is a programming language which is mainly known as the programming language of the worldwide web. It is also used in handheld devices, different electronic devices, and a wide range of tools which use computer technology for doing their tasks. The major characteristic of this language is that it is system independent. It means that a code written in Java in a Microsoft Windows operating system environment will have the exact outcome and looks, in any other platform running under different operating systems. These benefits were the main reason for using Java as the programming language in this thesis. The details about the reason for manipulating the exported XML document and the details about this process are provided in Section 6.4.4. The final XML document can be imported into the database.

Microsoft Access™ was selected as the database management system because of its wide use, easy application, and its availability in most of the real world situations. The author of the thesis developed a database for the data gathered by the forms. The database is introduced in Section 6.4.1.

In the following sections, different aspects of the Smart Form template developed by the author are explained.
6.3. THE PRINCIPLES IN DESIGNING FORMS

As mentioned earlier, the forms used as a case study for this thesis were elevated into electronic forms by the author of the thesis. For having an acceptable design and look for the forms a brief research was done to consider the best recommendations available for designing forms which are presented in this section.

Like every kind of design, there are some basic design principles for fillable forms which can help the designer, although there is not any official standard for this purpose. Each organization or company may have developed some guidelines to keep its forms efficient, consistent and manageable. One of the main factors is the appearance of the form. The form designer has more freedom if it is not supposed to be printed. In this case, different background photos and textures can be used, and there is no need for considering page limitations. The Business Forms Management Association (BFMA, www.bfma.org/) has been working for 45 years on form elements and provides useful recommendation for designing acceptable forms. Some of the important topics which are addressed by BFMA are Positioning, Grouping Form Elements, Fonts and Rules, Use of White Space, Language, Check Boxes/Radio Buttons, and so on (T. Padova, A. Okamoto, 2009).

The World Wide Web Consortium (W3C) has also developed a standard and an XML schema, XForms, for developing web forms. For several years, W3C has been working on XForms to make a standard set of XML forms to be used in a wide range of devices from PC to mobile, and other systems.
6.4. Different Parts of the Smart Form Template Developed for This Thesis

The idea of a *Construction Smart form* is not only limited to the fillable form, but is considered as a system which starts with the pdf fillable forms and ends up in a central data repository. The system may use different software applications and IT systems to fulfill the functions expected. These applications contain the data storage system and the method by which data can be transferred from the forms. In the following the database system developed for this system is introduced.

6.4.1. ACRP Report 21 Database

Investigating data gathered by the forms and finding the functional dependencies among them, showed the relational characteristics of the data, this resulted in choosing the database solution for storing data over a spreadsheet. The data from the forms were normalized and resulted in the following 23 tables, from which 16 tables are filled in with data from the first form, *Part 1-General Specifications*; three tables are related to the second one, *Analytical Approach form*, and four tables keep data from *Part 2-Weighted Matrix Approach form*.

The tables designed for capturing data from Part 1-General Specifications are as follow:

```
GeneralSpecifications (ProjectName, ProjectLocation, ReqDeliveryDate, EstDeliveryPeriod, EstProjectBudget, PaybackPeriod, ROICapital)
RiskSources (ProjectName, ItemNo, RiskSource)
SecurityIssues (ProjectName, ItemNo, SecurityIssue)
MajorChallenges (ProjectName, ItemNo, MajorChallenge)
OtherRiskIssues (ProjectName, ItemNo, OtherRisks)
MileStones (ProjectName, ItemNo, Milestone, MilestoneDate)
```
Below, there is the list of tables designed to capture data gathered in Tier1- Analytical approach form:

```
<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tier1Selections</td>
<td>(ProjectName, Issue, PDM, CaseNo, advantage)</td>
</tr>
<tr>
<td>AdvantageSummary</td>
<td>(ProjectName, Issue, PDM, Score)</td>
</tr>
<tr>
<td>Tier1Conclusion</td>
<td>(ProjectName, Conclusion)</td>
</tr>
</tbody>
</table>
```

The tables which contain data from Tier2- Weighted Matrix approach are as follows:

```
<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SelectionFactor</td>
<td>(ProjectName, SelectionFactor, Description, FactorWeight)</td>
</tr>
<tr>
<td>PDMWeights</td>
<td>(ProjectName, SelectionFactor, PDM, Score)</td>
</tr>
<tr>
<td>TotalScores</td>
<td>(ProjectName, PDM, TotalScore)</td>
</tr>
<tr>
<td>Tier2Conclusion</td>
<td>(ProjectName, Conclusion)</td>
</tr>
</tbody>
</table>
```

In this database, the GeneralSpecifications table has a primary key, ProjectName, which is used in all other tables as the foreign key. It results in a specific kind of database schema which is known as Star Schema (Kroenke & Auer, 2010). Figure 6.2 displays the Star schema of ACRP Report 21 database. Where GeneralSpecifications is the Fact Table and other tables radiating from it, are called Dimension Tables. The numbers beside the tables shows that there is a 1:N relationship between GeneralSpecifications table and other tables in the database.
6.4.2. Managing Form Data

Data are typically gathered for a future purpose. Mostly in all situations data is needed to be stored for more processing. For the purpose of this thesis, data are supposed to be saved in a database to be used for future research in finding the trends in selecting the project delivery method in Airport projects. In the previous section, the principles
used in designing the forms and the related database were explained. Designing the connection path between these two points is the next step which can be considered the part that distinguishes a Smart Form from a regular form. Otherwise it will be the same old method of retyping the information into other software application or database systems. Different methods can be applied to make this transition happen. Selecting the appropriate method depends on different factors such as complexity of data, access to IT experts, the budget available and the computer facilities available at locations where forms are used. In the following these methods are introduced.

**Methods for delivering the forms to the recipients**

Following methods can be considered for delivering forms to the recipients:

- Forms can be hosted by a website. Then they can be downloaded, filled and sent back by email or be filled online and submitted directly. This method does not need a specific IT knowledge. The only skill is to know how to work with an email provider system. This method is prefered for small AEC projects which does not have a developed software infrastructure to support more automatic methods of data transfer.

- Another way is sending forms directly to the recipients by email which can be done by saving them and attaching to an outgoing email or using the distribute option in Adobe Acrobat. The benefits of the second method are described later.

**Receiving responses from the forms hosted on a website**

Different methods are also available for receiving filled forms. Receiving a filled form can be easily accomplished by having it attached to a received email, or using a
server product for collecting responses. When using the server products, the following options are available (T. Padova, A. Okamoto, 2009):

- The forms can be submitted into a network folder on an internal company server or the form developer’s computer.

- The forms can be designed to be submitted to a URL (Uniform Resource Locator). This is the most straight-forward way for receiving responses, and needs help from IT department to program the server. This method might not be a choice for small companies and projects.

- Another method is using services from third party vendors for collecting, and possibly routing, the responses. One example is using the free service from Acrobat.com, which is introduced in the following part.

**Using Distribute and Tracker features in Adobe Acrobat™**

One method for sending the forms is to use the Distribute option available in Acrobat. It is recommended to use Distribute option regardless of the fact that the form is being sent directly or attached in an email. (T. Padova, A. Okamoto, 2009). Because it will automatically add the form to the Tracker feature and produces a response file to collect the responses. The Distribution feature offers three methods: (1) automatically download and organize forms by acrobat.com, (2) automatically collect responses in internal server, and (3) manually collect responses in email inbox.

*formName_distributed.pdf and formName_responses.pdf*
one way to gather the data entered in a form is using the service provided by third party vendors. Adobe was introduced as a provider of these services and the service was introduced, which is called Distribute and Track features.

Using Distribute option in Acrobat creates two pdf files. One is `formName_distributed.pdf`. This is the final pdf form which should be sent to the form recipients in one of the methods mentioned before. `formName_responses.pdf` is a portfolio pdf file which collects data from response files and arranges it in a spreadsheet format. The two files are linked to each other, so editing `formName_distributed.pdf` may disconnect this link. If Adobe.com is used for providing service, the responses file will be updated automatically from the website.

When manual email receiving is chosen, the response portfolio provides the feature for manually adding newly received responses to its spreadsheet. The information can be exported to a comma-separated values or character-separated values (CSV) simple text format file or Excel spreadsheet format. Figure 6.3 shows the `GeneralSpecifications_responses.pdf` which contains data from 5 responses.

![Figure 6.3. The GeneralSpecifications_response.com Portfolio Shows Five Responses in a Spreadsheet Format](image)
The Tracker feature in Adobe Acrobat

Adobe Acrobat provides a tool for managing the responses, from inside the application. Track Forms option in the Form menu of Acrobat keeps track of responses for the forms which were deployed using the Distribute option. Figure 6.4 shows a snapshot of Tracker which is keeping track of five different forms. As seen in this figure, the selected has five responses and one of the recipients has not responded yet.

![Tracker Keeps Track of the Responses from the Forms Which Were Deployed by Distribute Feature](image)

Using Acrobat.com service

As mentioned before, using online services from third party form service providers is an alternative way for programming the server, when forms are being managed via servers. Acrobat.com is one of these services, which is freely available by registration. It can work like an email server which receives the responses and automatically updates the responses files on the computer of the form owner.
6.4.3. Using XML Technology

In different parts of this thesis XML technology was introduced as the core technology which makes the idea of Smart Form possible. One of the benefits of Livecycle Designer™ is its capability in working with XML. It has a built-in set of XML elements which can completely define a form in XML format. This XML data can be exported into a .xdp file which Livecycle can make the pdf form from it. Figure 6.5 shows a Text Field element and its corresponding XML data in its corresponding .xdp file created by Livecycle. This feature provides much flexibility in manipulating the forms, by making it manageable in the widely used XML format.

Livecycle provides two ways for connecting XML to the form, connecting to a XML schema or to a Sample XML Data. Alternatively, the form designer may not need to connect any XML source to the form and take the desired data by defining the hierarchy and names of the elements in an appropriate way to use the built-in XML architecture, and using the built-in XML capabilities to export the data in an XML document.
Using built in XML features

As shown in Figure 6.5, every element in a pdf form has a corresponding XML element. Figure 6.5.a shows a TextField element in a Livecycle Designer™ form and Figure 6.5.b shows its corresponding xml definition. This feature can be used to extract data from the form into an XML document without the need for connecting an XML source to it. For example, assume that the Text Field in Figure 6.5 is filled with
Logan Airport Expansion I. The exported XML document will contain the following element:

\[\text{<ProjectName>Logan Airport Expansion I</ProjectName>}\]

By using this method, the only concern for the form designer is to arrange the hierarchy of elements in accordance with the desired data structure and use the appropriate names for the elements in the form.

**Setting the elements in ACRP Report 21pdf form in harmony with the database**

The tables and fields of ACRP Report 21 database were listed in Section 6.4.1 and shown in Figure 6.2. The XML representation of a record of a table contains an element with the same name as the table and subelements for the fields of that record which must also have the same tag as fields’ names. For example *RiskSources* table in the ACRP Report21database will have the following XML tree structure:

```
Element: <RiskSources>

Sub-Element: <ProjectName>
Sub-Element: <ItemNo>
Sub-Element: <RiskSource>
```

*Figure 6.6. Tree Model of Risksources Table and Its Fields*

In Figure 6.6, Table RiskSources is an element in the xml document and its fields are represented by children elements (sub-elements) in the corresponding xml document.

Figure 6.7 shows a sample of one of the elements in the form and its Hierarchy. Elements *Table* and *HeaderRow* have a binding property of *None*, which eliminates
them from the XML data file.

This hierarchy results in the following XML data when exported:

```
<MileStones>
  <MileStone>Start of Project</MileStone>
  <MilestoneDate>2010-09-06</MilestoneDate>
</MileStones>
```

ACRP Report 21 database can import this XML code and add the values in MileStone and MilestoneDate into a new record of MileStones table. Importing the data in this step leaves two other fields blank. One is ItemNo and the other one is ProjectName and they make a composite primary key for MileStones table. ItemNo is set to be automatically added in the database. But filling the ProjectName field with appropriate value will be explained later.
6.4.4. Modifying the XML Data File by DOM and Java

In many situations the XML data file exported from an application needs some modification before it is fully capable of providing data for another application. It can be done using XML Stylesheet Language Transformation (XSLT) or by manipulating the document using a DOM Application Programming Interface (API) in a programming language such as Java. Many programs have an option for choosing the appropriate XSLT file when importing an XML data file. In this project a free DOM API for java was used to manipulate the XML document.

Modifications by Java and DOM

The XML data file from the forms needs to be improved to be fully importable into the database:

- Some data entered in the form contain multiline texts which will be located in one field in the database. Each line of these data might need to be in a separate row of the table for future analysis. This needs some process to separate them to an acceptable format.

- The database contains records from different projects. ProjectName field keeps the information of each project unique and is the connector key of the records of a project. It must be added as a sub-element with ProjectName tag to all the elements in the XML document before importing it into the database.

In the first step, the Java code divides multiline data into several elements (SeparateLine.jar in Figure 6.1). For example consider the following example:

```
< RiskSources>
  < RiskSource>
    Snow storm & Site conditions
  </ RiskSource>
</ RiskSources>
```
In the XML data above, \&#xD; shows a carriage return. The above data will be changed to the following lines by the code:

```xml
<RiskSources>
  <RiskSource>Snow storm</RiskSource>
  <RiskSource>Site conditions</RiskSource>
</RiskSources>
```

In the first step in modifying the document, two lines of RiskSource were separated into two sub-elements each containing one line of data. But if this data are imported into the database, only the first line will be added in the RiskSource field of a new record of RiskSources table. The reason is that this set of data shows one record of RiskSources table which logically cannot contain two values for one field. Any duplicate field in a table needs to be in a new record in that table. The second module of designed Java code creates new elements for all the instances of a sub-element inside an element, and moves the duplicates into the new elements, until no element in the XML document contains two or more sub-elements with the same tag (reviseDuplicate.jar in Figure 6.1). In our example, the result will be like this:

```xml
< RiskSources>
  < RiskSource>Snow storm</RiskSource>
</RiskSources>

< RiskSources>
  < RiskSource>Site conditions</RiskSource>
</RiskSources>
```
Now this piece of XML data will create two new rows in RiskSources and adds both fields into the database. But one step is left. How the database is supposed to find out which rows in a table are related to which project? There is a ProjectName field for each row in the table, but the above dataset does not contain any data to fill that field. The third developed code deals with this problem (addKeyField.jar in Figure 6.1). It adds the foreign key to all the tables in the XML file which do not have the foreign key. After this stage the XML file of our example can be imported to the database:

```xml
<RiskSources>
  <ProjectName>Logan Airport</ProjectName>
  <RiskSource>Risk 1</RiskSource>
</RiskSources>

<RiskSources>
  <ProjectName>Logan Airport</ProjectName>
  <RiskSource>Risk2</RiskSource>
</RiskSources>
```

After performing the above-mentioned process on the XML data file, it will be ready to be added to the database.

### 6.5. Validation of The System

Different parts of the Smart Form system which was particularly developed for this thesis were explained in the previous sections of this chapter. Like every model, this system needs to be validated. Appendix B shows an ACRP Report 21 form which is
populated by sample data. The three xml documents resulting from the process in Figure 6.1. are presented. Also the Java codes are added to the appendix.

6.6. THE CONTRIBUTION OF THIS THESIS IN DEVELOPING THE IDEA OF CONSTRUCTION SMART FORMS SYSTEM

The main characteristic of a Construction Smart Form is its contribution in increasing data interoperability and seamless data transformation. Having this viewpoint, ACRP Report 21 system can be considered a Smart Form which transfers data from fillable forms to the database. As mentioned before in different parts of this thesis, different methods and tools can be used to fulfill this task and we considered developing a solution which would be affordable and efficient for most contractors and in nearly all situations. This system uses the tools which are widely and easily available. The forms can be saved and carried by any form of storage media such as flash disks or email and the data can be easily extracted and added to the database.
CHAPTER 7. CONCLUSION

A sample system was developed to test some characteristics of Construction Smart forms in practice. An available set of forms was improved to fillable dynamic pdf forms to provide a convenient environment for form recipient. Data interoperability between the form and the final destination of data is considered to be the most important factor for a Smart form to reduce time and error caused by retyping data. A database was designed for storing the data gathered by the forms. Information Technology allows using different solutions for transferring data from the forms to their destinations. Each method has its benefits and shortcomings which makes them desirable in different situations. The method of harmonizing the data container elements of the form with the database was chosen as the first attempt in creating Smart Forms due to its least dependency on specific IT knowledge. Three Java modules were designed using DOM XML parser to make new XML documents fully compatible with the database of the exported documents, as explained in Chapter 6. The first module, SeparateLines.jar, separates multiline values in an xml element to several same elements each containing one line. The second module, reviseDuplicate.jar, makes new table elements from an element which contains more than one identical sub-element and puts those sub-elements into new elements to fill in different rows in the same table in the database. The third module, addKeyField.jar, adds the key field of the tables as a sub-element with the appropriate value to the elements in the xml document. The information which is gathered in the
database will be used for further research and statistical computations for historical
trend analyses and other characteristics in selecting the most appropriate delivery
system.

A Construction Smart form can contain more features to be even more capable of
improving the efficiency of data transfer. These features are proposed, but not limited
to, as follow:

- There are XML schemas available for AECOO industry, which are globally
  accepted as the standard data structures. The form designer must investigate the
  possibility of using these standards in designing the XML schemas for his forms
  as much as possible. This makes the forms usable for more cases and puts them
  right in the stream of BIM of a project, ready to talk to other applications.

- Many forms may need some general data or calculations which can be provided in
  the form by using Web Services Definition Language (WSDL). As an example,
  web services can be used for obtaining the necessary information about
  construction products from web-based construction products catalogs. Livecycle
  Designer™ allows the form to connect to WSDL services. An example could be
  updated weather condition that can be automatically added to the form, or using
  currency converter in the form.

- The use of internet and hand-held devices will definitely have considerable effects
  on data exchange in a construction project. Online HTML or XHTML forms are
  the suitable interfaces for capturing data in this situations.

- Rule-Checking is another advanced capability which can be added to the form.
  This is a new area and still under development. Different rules, regulations, codes,
and practices can be controlled and the form may provide appropriate advice for different situations using rule-based techniques.

New advances in Information and communication technology will provide new tools for developing more advanced solutions for interoperability, while the more widespread accessibility will make the IT solution available for more ranges of AEC/FM firms.
APPENDIX A - REFERENCES


buildingSMART alliance. (2008). agcXML Common Definitions-Issued for final review by the agcXML review and validation committee. buildingSMART alliance (bSa).


Quin, L. R. (2010). *XML Essentials*. Retrieved from W3C:
   http://www.w3.org/standards/xml/core


   http://www.w3schools.com/wSDL/default.asp

   http://www.w3schools.com/dom/default.asp

   http://www.w3schools.com/xml/default.asp


Wix, J. (2007). *What is IFC?* Retrieved from buildingSMART Nordic:
APPENDIX B. CODES AND SAMPLE RESULTS

The three Java codes developed for this thesis are provided here. Then a sample filled out form of Part1_Specifications is showed with its exported XML document, logan.xml. This document has been processed in three processes, as described in Chapter 6, and the resulted xml documents are presented in order.
PART 1. JAVA CODES
/**This code separates the multi line value of a field in the input xml file to new elements of that kind. It is customized for the xml file exported from Part1_Specification.pdf, but can be modified for any other xml document**

import acm.program.*;
import java.io.*;
import javax.swing.*;

public class separateLines extends ConsoleProgram {

    public static void main(String[] args){
        new separateLines().start(args);
    }

    public void run(){
        println("This program converts multi line elements into several one line (one issue)elements with the same Tag");
        println("This program uses acm.jar library from Stanford University");
        println("*** Note: this version is customized for the xml file exported from Part1_Specification.pdf***");
        println("***Babak J.Fard, Northeastern University, July 2010****");
        println("Select the input file:");
        String rawText=null;
        try{
            rawText= readWithStringBuffer(openFileReaderUsingChooser());
        } catch (IOException e){
            println("Error in reading the file!");
        }
        String resultText="";
        Flag=0;
        while (rawText.indexOf(ELEMENT_DELIMITER,Flag)!=-1){
            resultText+=decompose(rawText.substring(Flag), ELEMENT_DELIMITER, START_SYMBOL, END_SYMBOL);
        }
        resultText+=rawText.substring(Flag);
        println("***XML file revised***");
        println("Input File: "+inputFile);
        String OutputFileName= inputFile.substring(0, inputFile.lastIndexOf(
            "."))+"_r"+
        +inputFile.substring(inputFile.lastIndexOf(".")));
        println("Output File: "+OutputFileName);
        WriteToFile(OutputFileName,resultText);
    }

    private String decompose(String statement, String delimiter, String startSymbol, String endSymbol){
        int delimiterPlace=statement.indexOf(delimiter);
        int startOfElement=reverseIndexOf(statement, delimiterPlace, startSymbol);
        int endOfElement=statement.indexOf(endSymbol, delimiterPlace);
        String ElementName=statement.substring(startOfElement+1,statement.
indexOf(">", startOfElement) );
String XMLElement=""+ElementName+"">
String XMLElementClosing=""+ElementName+"">
String ElementData=statement.substring(startOfElement+XMLElement.
length(),endOfElement);
String copyString=statement.substring(0, startOfElement);
Flag+=endOfElement+XMLElement.length()+1;
String revisedElement=splitElement(ElementData, delimiter,XMLElement,
XMLElementClosing);
return copyString+revisedElement;

/* This method distinguishes an element in a given text, which is surrounding
a given point in the text.
* the start and end symbols are given to the method to distinguish and
extract the element to a new String
*/
public String getElement (String inputText, int position, String
startSymbol, String endSymbol){
  String result=inputText.substring(startOfElement(inputText, position,
startSymbol),
      endOfElement(inputText, position, endSymbol));
  return result;
}

/* This method returns the start position of an element which is
containing a defined point in the
* text. the input parameters are the text, the place of the pointer in
the text, and the start symbol
* by which, the method can find the start point of the element.
*/
private int startOfElement(String inputText, int position, String
startSymbol){
  int result=reverseIndexOf(inputText, position, startSymbol);
  return result;
}

/* This method finds the end point of an element in the text.
*/

private int endOfElement(String inputText, int position, String
endSymbol){
  int result= inputText.indexOf(endSymbol, position);
  return result;
}

/*This method returns the index of the first accidence of the string in
the reverse order, or 0 and below
* if it does not appear.
*/
private int reverseIndexOf(String statement, int position, String s){
  while (position>0){
    position-=s.length();
    String CompareString=statement.substring(position, position+ s.
    length());
    if (CompareString.equals(s)) break;
  }
  return (position>=0 ? position: -1);
}

/* This method divides the text element at the delimiters. the result is
several same elements. If
* there are n delimiters in the element, (n-1) same elements will be
derived from the element.
* The input must be the data of an element, for an xml element the
StartSymbol is <ElementName>
* and EndSymbol is considered </ElementName>
*
*/
private String splitElement(String ElementData, String delimiter, String StartSymbol, String EndSymbol){
    String ElementName=StartSymbol.substring(StartSymbol.indexOf("\nSTART_SYMBOL\n")+1, StartSymbol.indexOf("\n"));
    if (isInList(TO_DICOMPOSED,ElementName)){
        int Flag=0;
        String revisedElement="";
        int LineNumber=1;
        while (ElementData.indexOf(delimiter,Flag) != -1) {
            String newElement=ElementData.substring(Flag, ElementData.indexOf(delimiter, Flag));
            revisedElement+=(StartSymbol+newElement+EndSymbol);
            Flag= (ElementData.indexOf(delimiter, Flag)+delimiter.length());
            LineNumber++;
        }
        if (LineNumber !=1){
            println("Element "+ElementName" : Broke into "+LineNumber+" elements!");
        }
        revisedElement+=(StartSymbol+ElementData.substring(Flag)+EndSymbol);
        return revisedElement;
    } else {
        String revisedElement=StartSymbol+ElementData+EndSymbol;
        return revisedElement;
    }
}

/* this method makes possible using a File Open dialogue Box, so the user can
browse for and select
* the required file. the method returns an open BufferedReader associated
with the selected file, but
* only if the user clicks open. adapted from the Book "The Art and Science of
Java"
*/
private BufferedReader openFileReaderUsingChooser(){
    BufferedReader rd=null;
    JFileChooser chooser=new JFileChooser();
    int result=chooser.showOpenDialog(this);
    if(result==JFileChooser.APPROVE_OPTION){
        try{
            File file=chooser.getSelectedFile();
            rd=new BufferedReader(new FileReader(file));
        }catch(IOException ex){
            println("can't open that file.");
        }
    }
    inputFile=chooser.getSelectedFile().getPath();
    return rd;
}

/*this method reads the contents of a reader into a StringBuffer and returns
the result into a String.
* This way is much faster than using the String at the first hand and add the
lines into it.
* adapted from the Book "The Art and Science of Java"
*/
static String readWithStringBuffer(Reader fr) throws IOException {
    BufferedReader br = new BufferedReader(fr);
String line;
StringBuffer result = new StringBuffer();
while ((line = br.readLine()) != null) {
    result.append(line);
}
return result.toString();

/**
 * This method reads the contents of a reader into a String.
 * This way is much faster than using the String at the first hand and add the
 * lines into it.
 * adapted from the Book "The Art and Science of Java"
 */

static String readWithString(Reader fr)
    throws IOException {
    BufferedReader br = new BufferedReader(fr);
    StringBuffer result = new StringBuffer();
    while ((buffer = br.readLine()) != null) {
        result.append(buffer);
    }
    return result.toString();
}

/* This method makes a new file and writes the desired String into the file
* adapted from the Book "The Art and Science of Java"
*/

private void WriteToFile(String FileName, String text) {
    try {
        PrintWriter wr = new PrintWriter(new FileWriter(FileName));
        wr.print(text);
        wr.close();
    } catch (IOException ex) {
        println("error writing the file: "+FileName);
    }
}

/* This method checks if a value, here a String, is in a list of values.
*/

private boolean isInList(String[] inputList, String Value) {
    for (int i=0; i<inputList.length; i++) {
        if (Value.equalsIgnoreCase(inputList[i])){
            return true;
        }
    }
    return false;
}

private int Flag;
private String inputFile;
// The list of elements which must be decomposed into new elements if contain
// multi line values.
private static final String[] TO_DICOMPOSED = {"RiskSource", "SecurityIssue",
    "MajorChallenge",
    "OtherRisks", "FundSource", "MajorFeature", "MajorStakeholder",
    "ProjectManager",
    "SustainabilityGoal",
    "OtherGoal"};

private static final String START_SYMBOL="<";
private static final String ELEMENT_DELIMITER="&amp;xD;";
private static final String END_SYMBOL=""/";
import java.io.*;
import javax.swing.*;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import javax.xml.transform.*;
import javax.xml.transform.stream.*;
import javax.xml.transform.dom.*;

public class reviseDuplicateFields {

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("This program separates several sub-elements with the same tag into new elements to be compatible with the format readable by a DBMS");
        System.out.println("This program uses acm.jar library from Stanford University");
        System.out.println("*** Note: this version is customized for the xml file exported from Part1_Specification.pdf***");
        System.out.println("***Babak J.Fard, Northeastern University, July 2010***");
        System.out.println("Select the input file:");
        XMLDoc = LoadXMLDocument();
        for (String dupl: DUPLICATE_LIST){
            NodeList elements = XMLDoc.getElementsByTagName(dupl);
            for (int i=0; i<elements.getLength(); i++){
                Node Element = elements.item(i);
                replaceDuplicateSubElements(Element);
            }
        }
        String outputFileName = inputFile.substring(0, inputFile.lastIndexOf("."))+"_d"+inputFile.substring(inputFile.lastIndexOf("."));
        writeXMLFile(XMLDoc, outputFileName);
    }

    /*
     * @param args
     */
    static Document LoadXMLDocument(){
        File file = openFileReaderUsingChooser();
        Document doc = null;
        try{
            DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
            DocumentBuilder db = dbf.newDocumentBuilder();
            doc = db.parse(file);
```java
} catch (Exception e) {
    e.printStackTrace();
} return doc;

/* */
static File openFileReaderUsingChooser() {
    File file = new File("/");
    JFileChooser chooser = new JFileChooser();
    int result = chooser.showOpenDialog(null);
    if (result == JFileChooser.APPROVE_OPTION) {
        file = chooser.getSelectedFile();
    }
    inputFile = chooser.getSelectedFile().getPath();
    return file;
}

/* This is the main method which separates several same sub elements, and adds each one to */
/* a new element. It is necessary for making the XML file compatible with the XML format */
/* of the rows in a database table */
private static void replaceDuplicateSubElements(Node ElementNode) {
    NodeList ChildNodes = ElementNode.getChildNodes();
    if (ChildNodes.getLength() >= 2) {
        System.out.println("****Start****");
        for (int i = 0; i < ChildNodes.getLength() - 1; i++) {
            System.out.println("i = "+i);
            Node FirstElem = ChildNodes.item(i);
            if (FirstElem.getNodeType() == Node.ELEMENT_NODE) {
                int[] removeList = new int[ChildNodes.getLength()];
                for (int j = i + 1; j < ChildNodes.getLength(); j++) {
                    System.out.println("j = "+j);
                    Node SecondElem = ChildNodes.item(j);
                    if (SecondElem.getNodeType() == Node.ELEMENT_NODE) {
                        if (FirstElem.getNodeName().equals(SecondElem.getNodeName())) {
                            Node NewElement = ElementNode.cloneNode(false);
                            Element tobeAdded = (Element) SecondElem.cloneNode(true);
                            NewElement.appendChild(tobeAdded);
                            XMLDoc.getDocumentElement().appendChild(NewElement);
                            //debug
                            System.out.println("Element j: "+SecondElem.getNodeName());
                            ElementNode.removeChild(SecondElem);
                            removeList[j] = 1;
                        }
                    }
                }
            }
        }
        System.out.println("ChildNodes length:" + ChildNodes.getLength());
        System.out.println("Real Child Numbers: "+ ElementNode.getChildNodes().getLength());
    }
}
```
// This method writes a DOM document to a file. taken from http://www.exampledepot.com/egs
public static void writeXMLFile(Document doc, String filename) {
    try {
        // Prepare the DOM document for writing
        Source source = new DOMSource(doc);

        // Prepare the output file
        File file = new File(filename);
        Result result = new StreamResult(file);

        // Write the DOM document to the file
        Transformer xformer = TransformerFactory.newInstance().newTransformer();
        xformer.transform(source, result);
    } catch (TransformerConfigurationException e) {
    } catch (TransformerException e) {
    }
}

private static Document XMLDoc=null;
private static String inputFile;
/* This code is to add the key sub element to the XML elements of a XML file. 
The idea is to 
  add the key fields to the tables of a database. the input file is a XML 
  which has elements of a group of 
  database tables. the code adds key attribute to the tables which doesn't 
  have it. these tables are introduced by 
  a series of Strings. 
  
  further improvements for future: at the moment the code may not work 
  properly for the situations when the name 
  of an attribute in a table is the same as the name of the table itself. for 
  improvement, the code must 
  distinguish the table from attributes. this way the code will only make 
  changes inside the table, not inside 
  the attributes. 
  
  this code works on extracting the names of all XML element, every text 
  between '<' and '>' characters, and 
  compares them to the eligible names. 
  */

import acm.program.*;
import java.io.*;
import javax.swing.*;

public class AddKeyElement extends ConsoleProgram {
  int Flag=0;
  String inputText="";

  public static void main(String[] args){
    new AddKeyElement().start(args);
  }

  public void run(){
    println("This program adds the key Fields to the appropriate elements 
    which represent the rows of the tables in the database");
    println(" This program uses acm.jar library from Stanford 
    University+ XML DOM parser from w3c.org");
    println(" *** Note: this version is customized for the xml file 
    exported from Part1_Specification.pdf***");
    println(" ***Babak J.Fard, Northeastern University, July 
    2010***");

    try{
      inputText= readWith StringBuffer(openFileReaderUsingChooser());
    } catch (IOException e){
      println("Error in reading the file!");
    }

    int StartofKey=(inputText.indexOf("<"+KEY_ELEMENT+">"));
    int EndofKey=(inputText.indexOf("</"+KEY_ELEMENT+">")+KEY_ELEMENT. 
    length()+3);
    String keySubElement=inputText.substring(StartofKey,EndofKey);
    println ("The key field to be added: "+keySubElement);

    while (Flag!=1){
      String newElement= getElementName(inputText.substring(Flag));

      if (isInList(TO_DICOMPOSED,newElement)){
        println(" *****The key attribute was added to this table: "+ 

newElement+"******");
println();
inputText=addElement(inputText,"<"+newElement, keySubElement, Flag);
Flag=inputText.indexOf("<",Flag+1)+keySubElement.length();
if (Flag==-1) break;
}
else {
println("cursor Location: "+Flag);
println("**Table: "+newElement+. no need to add key attribute");
Flag=inputText.indexOf("<",Flag+1);
}
String OutputFileName= inputFile.substring(0, inputFile.lastIndexOf(".")+inputFile.substring(inputFile.lastIndexOf(".")));WriteToFile(OutputFileName,inputText);

/* This method returns the name of an XML element. the input is the whole element with its data and the output is the name of the element */
private String getElementName(String theElement){
    int StartIndex=theElement.indexOf("<");
    int EndIndex=theElement.indexOf(">");
    if (EndIndex<StartIndex){
        return "Not Valid Name";
    }else{
        return theElement.substring(StartIndex+1, EndIndex);
    }
}

/* This method checks if a value, here a String, is in a list of values. */
private boolean isInList(String[] inputList,String Value){
    for (int i=0;i<inputList.length;i++){
        if (Value.equalsIgnoreCase(inputList[i])){
            return true;
        }
    }
    return false;
}

/*This method constructs a valid XML element. it inputs the name of the element and its related data and joins them into a standard XML element */
private String SetXMLelement(String ElementName, String ElementData){
    String result="<"+ElementName+"">"+ElementData+"</"+ElementName+"">";
    return result;
}

/* this method makes possible to use a File Open dialogue Box, so the user can browse for and select the required file. the method returns an open BufferedReader associated
with the selected file, but
* only if the user clicks open
*/
private BufferedReader openFileReaderUsingChooser()
{
    BufferedReader rd=null;
    JFileChooser chooser=new JFileChooser();
    int result=chooser.showOpenDialog(this);
    if(result==JFileChooser.APPROVE_OPTION){
        try{
            File file=chooser.getSelectedFile();
            rd=new BufferedReader(new FileReader(file));
        }catch(IOException ex){
            println("can't open that file.");
        }
    }
    inputFile=chooser.getSelectedFile().getPath();
    return rd;
}

/*this method reads the contents of a reader into a StringBuffer and
returns the result into a String.
* This way is much faster than using the String at the first hand and
add the lines into it.
*/
static String readWithStringBuffer(Reader fr) throws IOException {
    BufferedReader br = new BufferedReader(fr);
    String line;
    StringBuffer result = new StringBuffer();
    while ((line = br.readLine()) != null) {
        result.append(line);
    }
    return result.toString();
}

/* This method makes a new file and writes the desired String into the
file
*/
private void WriteToFile(String FileName, String text){
    try{
        PrintWriter wr=new PrintWriter(new FileWriter(FileName));
        wr.print(text);
        wr.close();
    }catch (IOException ex){
        println("error writing the file: "+FileName);
    }
}

/*This method adds a text (key sub-element) to the desired point in the
string
*/
private String addElement(String inputString, String definingPoint,
String tobeAdded, int Start){
    int basePoint=inputString.indexOf("<", inputString.indexOf(
definingPoint,Start)+2);
    String result=inputString.substring(0, basePoint)+tobeAdded+
    inputString.substring(basePoint);
    return result;
}

private static final String[] TO_DICOMPOSED= 
{"RiskSources", "SecurityIssues", "MajorChallenges"};
private static final String KEY_ELEMENT="ProjectName";
private String inputFile;
PART2. A SAMPLE ‘PART1-SPECIFICATION’ FILLED FORM
# Project Description

## General Specifications

<table>
<thead>
<tr>
<th>Project Name</th>
<th>Boston Logan Intl. Airport</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>Boston MA</td>
</tr>
<tr>
<td>Required Delivery Date</td>
<td>Aug 21, 2010</td>
</tr>
<tr>
<td>Estimated Project Delivery Period</td>
<td>36 months</td>
</tr>
<tr>
<td>Estimated Project Budget</td>
<td>$4,560,000</td>
</tr>
<tr>
<td>Payback Period</td>
<td>60 months</td>
</tr>
<tr>
<td>Return on Capital Investment (%)</td>
<td>3.4</td>
</tr>
</tbody>
</table>

## Risk Related Issues

<table>
<thead>
<tr>
<th>Main Identified Sources of Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large numbers of ground service operations and equipment around them and the potential exposure of fire and smoke to terminal buildings</td>
</tr>
<tr>
<td>The large proximity of aircraft and aircraft fueling operations</td>
</tr>
<tr>
<td>Flammable/Combustible Gas and Liquid Containers</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Security Issues or Concerns</th>
</tr>
</thead>
<tbody>
<tr>
<td>accessibility to people outside the construction siteNYC Security 2 employees and materials entering to the site must be checked</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Major Challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>tight budget schedule</td>
</tr>
<tr>
<td>the shortage of skilled workers</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Other Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Other</td>
</tr>
</tbody>
</table>
Time Related Issues

<table>
<thead>
<tr>
<th>Milestone</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>Aug 14, 2010</td>
</tr>
<tr>
<td>Substantial Completion</td>
<td>Aug 26, 2011</td>
</tr>
<tr>
<td>final Completion</td>
<td>Nov 9, 2011</td>
</tr>
</tbody>
</table>

Financial Issues

Sources of Project Funding

- Federal Budget
- State taxes
- Private investors

Major Features of Work

- Adding two new take off lanes
- Adding two new gates
- Expanding the navigation system

Major Stakeholders

- Boston Logan Airport
- MBTA
- Federal Aviation Agency

Project Manager(s)

- Eric Clapton
- Jimmy Page
- Joe Satriani

Substantial Design and Construction Requirements

- Using BIM in design and construction phases
- Complying with international and national standards of airport construction
- Construction may not commence until AV Environmental Program staff have reviewed and approved the project
### Project Goals

| Schedule   | Schedule Log1  
|------------|----------------
|            | Schedule Log2  
|            | Schedule Log3  
|            | Schedule Log4  
| Cost       | Log Cost 1    
|            | Log Cost 2    
|            | Log Cost 3    
| Quality    | Log Quality 1  
| Sustainability | Sustain Log 1  
|            | Sust Log 2    
|            | Sustain Log#3 
|            | Sustain Log4  
|            | Sustain Log5  
| Other      | OtherGoals1   

Airport Project Delivery Selection System

Project Description
PART 3. XML FILES

Logan.xml

This is the exported xml file from the filled in form.
<?xml version="1.0" encoding="UTF-8"?>
<topmostSubform>
  <GeneralSpecifications>
    <ProjectName>Boston Logan Intl. Airport</ProjectName>
    <ProjectLocation>Boston, MA</ProjectLocation>
    <ReqDeliveryDate>2010-08-21</ReqDeliveryDate>
    <EstDeliveryPeriod>36 months</EstDeliveryPeriod>
    <EstProjectBudget>4560000.00000000</EstProjectBudget>
    <PaybackPeriod>60 months</PaybackPeriod>
    <ROICapital>3.4</ROICapital>
  </GeneralSpecifications>
  <RiskSources>
    <RiskSource>Large numbers of ground service operations and equipment around them and the potential exposure of fire and smoke to terminal buildings; The large proximity of aircraft and aircraft fueling operations
    & Flammable/Combustible Gas and Liquid Containers</RiskSource>
  </RiskSources>
  <SecurityIssues>
    <SecurityIssue>accessibility to people outside the construction site; NYC Security 2; employees and materials entering to the site must be checked</SecurityIssue>
  </SecurityIssues>
  <MajorChallenges>
    <MajorChallenge>tight budget; schedule; the shortage of skilled workers</MajorChallenge>
  </MajorChallenges>
  <OtherRisks>
    <OtherRisks>Other</OtherRisks>
  </OtherRisks>
  <MileStones>
    <Milestone>Start</Milestone>
    <MilestoneDate>2010-08-14</MilestoneDate>
  </MileStones>
  <FundSources>
    <FundSource>Federal Budget; State taxes; Private investors</FundSource>
  </FundSources>
  <MajorFeatures>
    <MajorFeature>adding two new take off lanes; adding two new gates; expanding the navigation system</MajorFeature>
  </MajorFeatures>
  <MajorStakeholders>
    <MajorStakeholder>Boston Logan Airport; MBTA; Federal Aviation Agency</MajorStakeholder>
  </MajorStakeholders>
  <ProjectManagers>
    <ProjectManager>Eric Clapton; Jimmy Page; Joe Satriani</ProjectManager>
  </ProjectManagers>
  <SubstDCReqs>
    <SubstDCReq>- Using BIM in design and construction phases; - complying with international and national standards of airport construction; - Construction may not commence until AV Environmental Program staff have reviewed and approved the project</SubstDCReq>
  </SubstDCReqs>
  <ScheduleGoals>
    <ScheduleGoal>Schedule Log1; Schedule Log2; Schedule Log3; Schedule Log4</ScheduleGoal>
  </ScheduleGoals>
  <CostGoals>
    <CostGoal>Log Cost 1; Log Cost 2; Log Cost 3</CostGoal>
  </CostGoals>
  <QualityGoals>
    <QualityGoal>Log Quality 1</QualityGoal>
  </QualityGoals>
  <SustainabilityGoals>
    <SustainabilityGoal>Sustain Log 1; Sust Log 2; Sustain Log#3; Sustain Log4; Sustain Log5</SustainabilityGoal>
  </SustainabilityGoals>
</topmostSubform>
<OtherGoal>OtherGoals</OtherGoal>

<MileStones>
  <Milestone>Substantial Completion</Milestone>
  <MilestoneDate>2011-08-26</MilestoneDate>
</MileStones>

<MileStones>
  <Milestone>final Completion</Milestone>
  <MilestoneDate>2011-11-09</MilestoneDate>
</MileStones>
Logan_r.xml

The following is the resulted file from executing separateLines.jar on logan.xml
<?xml version="1.0" encoding="UTF-8"?>
<topmostSubform>
<GeneralSpecifications>
<ProjectName>Boston Logan Intl. Airport</ProjectName>
<ProjectLocation>Boston, MA</ProjectLocation>
<ReqDeliveryDate>2010-08-21</ReqDeliveryDate>
<EstDeliveryPeriod>36 months</EstDeliveryPeriod>
<EstProjectBudget>4560000.00</EstProjectBudget>
<PaybackPeriod>60 months</PaybackPeriod>
<ROI Capital>3.4</ROI Capital>
</GeneralSpecifications>
<RiskSources>
<RiskSource>Large numbers of ground service operations and equipment around them and the potential exposure of fire and smoke to terminal buildings</RiskSource>
<RiskSource>The large proximity of aircraft and aircraft fueling operations</RiskSource>
<RiskSource>Flammable/Combustible Gas and Liquid Containers</RiskSource>
</RiskSources>
<SecurityIssues>
<SecurityIssue>accessibility to people outside the construction site</SecurityIssue>
<SecurityIssue>employees and materials entering to the site must be checked</SecurityIssue>
</SecurityIssues>
<MajorChallenges>
<MajorChallenge>tight budget</MajorChallenge>
<MajorChallenge>schedule</MajorChallenge>
<MajorChallenge>the shortage of skilled workers</MajorChallenge>
</MajorChallenges>
<OtherRiskIssues>
<OtherRisks>Other</OtherRisks>
</OtherRiskIssues>
<MileStones>
<Milestone>Start</Milestone>
<MilestoneDate>2010-08-14</MilestoneDate>
</MileStones>
<FundSources>
<FundSource>Federal Budget</FundSource>
<FundSource>State taxes</FundSource>
<FundSource>Private investors</FundSource>
</FundSources>
<MajorFeatures>
<MajorFeature>adding two new take off lanes</MajorFeature>
<MajorFeature>adding two new gates</MajorFeature>
<MajorFeature>expanding the navigation system</MajorFeature>
</MajorFeatures>
<MajorStakeholders>
<MajorStakeholder>Boston Logan Airport</MajorStakeholder>
<MajorStakeholder>MBTA</MajorStakeholder>
<MajorStakeholder>Federal Aviation Agency</MajorStakeholder>
</MajorStakeholders>
<ProjectManagers>
<ProjectManager>Eric Clapton</ProjectManager>
<ProjectManager>Jimmy Page</ProjectManager>
<ProjectManager>Joe Satriani</ProjectManager>
</ProjectManagers>
<SubstDCReqs>
<SubstDCReq>Using BIM in design and construction phases</SubstDCReq>
<SubstDCReq>-complying with international and national standards of airport construction</SubstDCReq>
<SubstDCReq>-Construction may not commence until AV Environmental Program staff have reviewed and approved the project</SubstDCReq>
</SubstDCReqs>
</topmostSubform>
<ScheduleGoals>
  <ScheduleGoal>Schedule Log2</ScheduleGoal>
  <ScheduleGoal>Schedule Log3</ScheduleGoal>
  <ScheduleGoal>Schedule Log4</ScheduleGoal>
</ScheduleGoals>

<CostGoals>
  <CostGoal>Log Cost 1</CostGoal>
  <CostGoal>Log Cost 2</CostGoal>
  <CostGoal>Log Cost 3</CostGoal>
</CostGoals>

<QualityGoals>
  <QualityGoal>Log Quality 1</QualityGoal>
</QualityGoals>

<SustainabilityGoals>
  <SustainabilityGoal>Sustain Log 1</SustainabilityGoal>
  <SustainabilityGoal>Sustain Log 2</SustainabilityGoal>
  <SustainabilityGoal>Sustain Log#3</SustainabilityGoal>
  <SustainabilityGoal>Sustain Log4</SustainabilityGoal>
  <SustainabilityGoal>Sustain Log5</SustainabilityGoal>
</SustainabilityGoals>

<OtherGoals>
  <OtherGoal>OtherGoals1</OtherGoal>
</OtherGoals>

<MileStones>
  <Milestone>Substantial Completion</Milestone>
  <MilestoneDate>2011-08-26</MilestoneDate>
</MileStones>

<MileStones>
  <Milestone>final Completion</Milestone>
  <MilestoneDate>2011-11-09</MilestoneDate>
</MileStones>
Logan_r_d.xml

The following is the resulted file from executing reviseDuplicate.jar on logan_r.xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<topmostSubform>
  <GeneralSpecifications>
    <ProjectName>Boston Logan Intl. Airport</ProjectName>
    <ProjectLocation>Boston, MA</ProjectLocation>
    <ReqDeliveryDate>2010-08-21</ReqDeliveryDate>
    <EstDeliveryPeriod>36 months</EstDeliveryPeriod>
    <EstProjectBudget>4560000.00000</EstProjectBudget>
    <PaybackPeriod>60 months</PaybackPeriod>
    <ROICapital>3.4</ROICapital>
  </GeneralSpecifications>
  <RiskSources>
    <RiskSource>Large numbers of ground service operations and equipment around them and the potential exposure of fire and smoke to terminal buildings</RiskSource>
  </RiskSources>
  <SecurityIssues>
    <SecurityIssue>accessibility to people outside the construction site</SecurityIssue>
  </SecurityIssues>
  <MajorChallenges>
    <MajorChallenge>tight budget</MajorChallenge>
  </MajorChallenges>
  <RiskSources>
    <RiskSource>Other</RiskSource>
  </RiskSources>
  <MileStones>
    <Milestone>Start</Milestone>
    <MilestoneDate>2010-08-14</MilestoneDate>
  </MileStones>
  <FundSources>
    <FundSource>Federal Budget</FundSource>
  </FundSources>
  <MajorFeatures>
    <MajorFeature>adding two new take off lanes</MajorFeature>
  </MajorFeatures>
  <MajorStakeholders>
    <MajorStakeholder>Boston Logan Airport</MajorStakeholder>
  </MajorStakeholders>
  <ProjectManagers>
    <ProjectManager>Eric Clapton</ProjectManager>
  </ProjectManagers>
  <SubstDCReqs>
    <SubstDCReq>-Using BIM in design and construction phases</SubstDCReq>
  </SubstDCReqs>
  <ScheduleGoals>
    <ScheduleGoal>Schedule Log1</ScheduleGoal>
  </ScheduleGoals>
  <CostGoals>
    <CostGoal>Log Cost 1</CostGoal>
  </CostGoals>
  <QualityGoals>
    <QualityGoal>Log Quality 1</QualityGoal>
  </QualityGoals>
  <SustainabilityGoals>
    <SustainabilityGoal>Sustain Log 1</SustainabilityGoal>
  </SustainabilityGoals>
  <OtherGoals>
    <OtherGoal>OtherGoals1</OtherGoal>
  </OtherGoals>
  <MileStones>
    <Milestone>Substantial Completion</Milestone>
    <MilestoneDate>2011-08-26</MilestoneDate>
  </MileStones>
</topmostSubform>
<MileStones>
  <Milestone>final Completion</Milestone>
  <MilestoneDate>2011-11-09</MilestoneDate>
</MileStones>

<RiskSources>
  <RiskSource>The large proximity of aircraft and aircraft fueling operations</RiskSource>
  <RiskSource>Flammable/Combustible Gas and Liquid Containers</RiskSource>
</RiskSources>

<SecurityIssues>
  <SecurityIssue>employees and materials entering to the site must be checked</SecurityIssue>
</SecurityIssues>

<MajorChallenges>
  <MajorChallenge>schedule</MajorChallenge>
  <MajorChallenge>the shortage of skilled workers</MajorChallenge>
</MajorChallenges>

<FundSources>
  <FundSource>State taxes</FundSource>
  <FundSource>Private investors</FundSource>
</FundSources>

<MajorFeatures>
  <MajorFeature>adding two new gates</MajorFeature>
  <MajorFeature>expanding the navigation system</MajorFeature>
</MajorFeatures>

<MajorStakeholders>
  <MajorStakeholder>MBTA</MajorStakeholder>
  <MajorStakeholder>Federal Aviation Agency</MajorStakeholder>
</MajorStakeholders>

<ProjectManagers>
  <ProjectManager>Jimmy Page</ProjectManager>
  <ProjectManager>Joe Satriani</ProjectManager>
</ProjectManagers>

<SubstDCReqs>
  <SubstDCReq>- complying with international and national standards of airport construction</SubstDCReq>
  <SubstDCReq>- Construction may not commence until AV Environmental Program staff have reviewed and approved the project</SubstDCReq>
</SubstDCReqs>

<ScheduleGoals>
  <ScheduleGoal>Schedule Log2</ScheduleGoal>
  <ScheduleGoal>Schedule Log3</ScheduleGoal>
  <ScheduleGoal>Schedule Log4</ScheduleGoal>
</ScheduleGoals>
<CostGoals>
  <CostGoal>Log Cost 2</CostGoal>
</CostGoals>

<CostGoals>
  <CostGoal>Log Cost 3</CostGoal>
</CostGoals>

<SustainabilityGoals>
  <SustainabilityGoal>Sust Log 2</SustainabilityGoal>
</SustainabilityGoals>

<SustainabilityGoals>
  <SustainabilityGoal>Sust Log#3</SustainabilityGoal>
</SustainabilityGoals>

<SustainabilityGoals>
  <SustainabilityGoal>Sustain Log4</SustainabilityGoal>
</SustainabilityGoals>

<SustainabilityGoals>
  <SustainabilityGoal>Sustain Log5</SustainabilityGoal>
</SustainabilityGoals>

<OtherGoals>
  <OtherGoal/>
</OtherGoals>
Logan_r_d_key.xml

The following is the resulted file from executing addKeyFields.jar on logan_r_d.xml.

This file is ready to be imported to the database
<topmostSubform>
  <GeneralSpecifications>
    <ProjectName>Boston Logan Intl. Airport</ProjectName>
    <ProjectLocation>Boston&#13;MA</ProjectLocation>
    <ReqDeliveryDate>2010-08-21</ReqDeliveryDate>
    <EstDeliveryPeriod>36 months</EstDeliveryPeriod>
    <EstProjectBudget>4560000.00000000</EstProjectBudget>
    <PaybackPeriod>60 months</PaybackPeriod>
    <ROICapital>3.4</ROICapital>
  </GeneralSpecifications>
  <RiskSources>
    <ProjectName>Boston Logan Intl. Airport</ProjectName>
    <RiskSource>Large numbers of ground service operations and equipment around them and the potential exposure of fire and smoke to terminal buildings</RiskSource>
  </RiskSources>
  <SecurityIssues>
    <ProjectName>Boston Logan Intl. Airport</ProjectName>
    <SecurityIssue>accessibility to people outside the construction siteNYC Security 2</SecurityIssue>
  </SecurityIssues>
  <MajorChallenges>
    <ProjectName>Boston Logan Intl. Airport</ProjectName>
    <MajorChallenge>tight budget</MajorChallenge>
  </MajorChallenges>
  <OtherRiskIssues>
    <ProjectName>Boston Logan Intl. Airport</ProjectName>
    <OtherRisks>Other</OtherRisks>
  </OtherRiskIssues>
  <MileStones>
    <ProjectName>Boston Logan Intl. Airport</ProjectName>
    <Milestone>Start</Milestone>
    <MilestoneDate>2010-08-14</MilestoneDate>
  </MileStones>
  <FundSources>
    <ProjectName>Boston Logan Intl. Airport</ProjectName>
    <FundSource>Federal Budget</FundSource>
  </FundSources>
  <MajorFeatures>
    <ProjectName>Boston Logan Intl. Airport</ProjectName>
    <MajorFeature>adding two new take off lanes</MajorFeature>
  </MajorFeatures>
  <MajorStakeholders>
    <ProjectName>Boston Logan Intl. Airport</ProjectName>
    <MajorStakeholder>Boston Logan Airport</MajorStakeholder>
  </MajorStakeholders>
  <ProjectManagers>
    <ProjectName>Boston Logan Intl. Airport</ProjectName>
    <ProjectManager>Eric Clapton</ProjectManager>
  </ProjectManagers>
  <SubstDCReqs>
    <ProjectName>Boston Logan Intl. Airport</ProjectName>
    <SubstDCReq>Using BIM in design and construction phases</SubstDCReq>
  </SubstDCReqs>
  <ScheduleGoals>
    <ProjectName>Boston Logan Intl. Airport</ProjectName>
    <ScheduleGoal>Schedule Log1</ScheduleGoal>
  </ScheduleGoals>
  <CostGoals>
    <ProjectName>Boston Logan Intl. Airport</ProjectName>
    <CostGoal>Log Cost 1</CostGoal>
  </CostGoals>
  <QualityGoals>
    <ProjectName>Boston Logan Intl. Airport</ProjectName>
    <QualityGoal>Quality Log 1</QualityGoal>
  </QualityGoals>
</topmostSubform>
<ProjectName>Boston Logan Intl. Airport</ProjectName>

<QualityGoal>Log Quality 1</QualityGoal>

<SustainabilityGoals>
  <ProjectName>Boston Logan Intl. Airport</ProjectName>
  <SustainabilityGoal>Sustain Log 1</SustainabilityGoal>
</SustainabilityGoals>

<OtherGoals>
  <ProjectName>Boston Logan Intl. Airport</ProjectName>
  <OtherGoal>otherGoals1</OtherGoal>
</OtherGoals>

<MileStones>
  <ProjectName>Boston Logan Intl. Airport</ProjectName>
  <Milestone>Substantial Completion</Milestone>
  <MilestoneDate>2011-08-26</MilestoneDate>
</MileStones>

<MileStones>
  <ProjectName>Boston Logan Intl. Airport</ProjectName>
  <Milestone>final Completion</Milestone>
  <MilestoneDate>2011-11-09</MilestoneDate>
</MileStones>

<RiskSources>
  <ProjectName>Boston Logan Intl. Airport</ProjectName>
  <RiskSource>The large proximity of aircraft and aircraft fueling operations</RiskSource>
</RiskSources>

<RiskSources>
  <ProjectName>Boston Logan Intl. Airport</ProjectName>
  <RiskSource>Flammable/Combustible Gas and Liquid Containers</RiskSource>
</RiskSources>

<SecurityIssues>
  <ProjectName>Boston Logan Intl. Airport</ProjectName>
  <SecurityIssue>employees and materials entering to the site must be checked</SecurityIssue>
</SecurityIssues>

<MajorChallenges>
  <ProjectName>Boston Logan Intl. Airport</ProjectName>
  <MajorChallenge>schedule</MajorChallenge>
</MajorChallenges>

<MajorChallenges>
  <ProjectName>Boston Logan Intl. Airport</ProjectName>
  <MajorChallenge>the shortage of skilled workers</MajorChallenge>
</MajorChallenges>

<FundSources>
  <ProjectName>Boston Logan Intl. Airport</ProjectName>
  <FundSource>State taxes</FundSource>
</FundSources>

<FundSources>
  <ProjectName>Boston Logan Intl. Airport</ProjectName>
  <FundSource>Private investors</FundSource>
</FundSources>

<MajorFeatures>
  <ProjectName>Boston Logan Intl. Airport</ProjectName>
  <MajorFeature>adding two new gates</MajorFeature>
</MajorFeatures>

<MajorFeatures>
  <ProjectName>Boston Logan Intl. Airport</ProjectName>
  <MajorFeature>expanding the navigation system</MajorFeature>
</MajorFeatures>

<MajorStakeholders>
  <ProjectName>Boston Logan Intl. Airport</ProjectName>
  <MajorStakeholder>MBTA</MajorStakeholder>
</MajorStakeholders>
<ProjectName>Boston Logan Intl. Airport</ProjectName>

<Federal Aviation Agency>

<SubstDCReqs>
  <ProjectName>Boston Logan Intl. Airport</ProjectName>
  <SubstDCReq>- complying with international and national standards of airport construction</SubstDCReq>
</SubstDCReqs>

<SubstDCReqs>
  <ProjectName>Boston Logan Intl. Airport</ProjectName>
  <SubstDCReq>- Construction may not commence until AV Environmental Program staff have reviewed and approved the project</SubstDCReq>
</SubstDCReqs>

<ScheduleGoals>
  <ScheduleGoal>Schedule Log2</ScheduleGoal>
</ScheduleGoals>

<ScheduleGoals>
  <ScheduleGoal>Schedule Log3</ScheduleGoal>
</ScheduleGoals>

<ScheduleGoals>
  <ScheduleGoal>Schedule Log4</ScheduleGoal>
</ScheduleGoals>

<CostGoals>
  <CostGoal>Log Cost 2</CostGoal>
</CostGoals>

<CostGoals>
  <CostGoal>Log Cost 3</CostGoal>
</CostGoals>

<SustainabilityGoals>
  <SustainabilityGoal>Sust Log 2</SustainabilityGoal>
</SustainabilityGoals>

<SustainabilityGoals>
  <SustainabilityGoal>Sustain Log#3</SustainabilityGoal>
</SustainabilityGoals>

<SustainabilityGoals>
  <SustainabilityGoal>Sustain Log4</SustainabilityGoal>
</SustainabilityGoals>

<SustainabilityGoals>
  <SustainabilityGoal>Sustain Log5</SustainabilityGoal>
</SustainabilityGoals>

<OtherGoals>
</OtherGoals>
<OtherGoal/>
</OtherGoals>
</topmostSubform>