

*A Description of Information System Technologies and Implementations of  
Project Information Management Systems for use in the South African  
Government*

Pierre van Zyl

Thesis presented in partial fulfilment of the requirements for the degree of  
*Master of Engineering (Civil)* at the University of Stellenbosch.



Study leader: JAvB Strasheim

December 2002

## **Declaration**

I, the undersigned, hereby declare that the work contained in this thesis, including all calculations, drawings, results, graphs and computer programmes are my own original work unless otherwise stated in the text and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature:

Date:



## Synopsis

This thesis focuses on information systems for project management in the South African Government with special attention to the Consolidated Municipal Infrastructure Programme. Project data exchange standards and the suitability of these standards for project information management in the South African Government is reviewed. Information system technologies applicable to project management, with reference to computer programming languages, markup languages, communication technologies, Internet technologies, database technologies and document manipulation tools are discussed. Project information management forms are then discussed, followed by an overview of the flow of project information during the lifecycle of a project. The standards development organisation, ActionIt, and the project information management models developed by it are highlighted. A description of applications and a system that was implemented to illustrate the work covered in this thesis is provided with examples of the applications and the system.

Hierna tesis fokus op inligting sisteme vir projekbestuur in die Suid Afrikaanse Regering met spesifieke verwysing na die Gekonsolideerde Munisipale Infrastruktuur Program. Standaarde vir die uitruil van projek inligting sowel as die toepaslikheid van hierdie standarde vir die bestuur van projek inligting in die Suid Afrikaanse Regering word bespreek. Informasie sisteem tegnologieë wat van toepassing is op die bestuur van projek inligting word behandel en sluit in: rekenaar programerings tale, 'markup' tale, kommunikasie tegnologieë, Internet tegnologieë, databasis tegnologieë en gereedskap om dokumente te manipuleer. Verskillende projek vorms word behandel gevolg deur 'n beskrywing van die vloei van data gedurende 'n projek se leeftyd. 'n Beskrywing van ActionIT, 'n organisasie vir die neerlê van standarde vir gebruik in die Suid Afrikaanse regering, sowel as die modelle wat deur ActionIT ontwikkel is vir die bestuur van projek inligting word gegee. Die toepassings en sisteem wat geïmplimenteer is om die werk wat in hierdie tesis behandel is te demonstreer word gegee met voorbeelde van die toepassings en sisteem.

## Acknowledgements

With this I want to thank the following people:

- Mr. JAvB Strasheim for his leadership and vision.
- My parents for their understanding and support.
- Rosemarie Moller who proofread this thesis.

## Table of contents

<b>DECLARATION .....</b>	<b>II</b>
<b>SYNOPSIS .....</b>	<b>III</b>
<b>ACKNOWLEDGEMENTS .....</b>	<b>IV</b>
<b>TABLE OF CONTENTS.....</b>	<b>V</b>
<b>LIST OF FIGURES .....</b>	<b>XII</b>
<b>LIST OF TABLES .....</b>	<b>XIV</b>
<b>CHAPTER 1.....</b>	<b>1</b>
<b>INTRODUCTION.....</b>	<b>1</b>
<b>CHAPTER 2.....</b>	<b>3</b>
<b>THESIS OBJECTIVES .....</b>	<b>3</b>
<b>CHAPTER 3.....</b>	<b>4</b>
<b>AN PROJECT MANAGEMENT SYSTEM IN THE SOUTH AFRICAN GOVERNMENT .....</b>	<b>4</b>
3.1 CONSOLIDATED MUNICIPAL INFRASTRUCTURE PROGRAMME (CMIP).....	4
3.1.1 <i>CMIP Management structure:</i> .....	5
3.1.2 <i>CMIP Functioning</i> .....	8
3.2 PER-FORM DEVELOPER .....	9
<b>CHAPTER 4.....</b>	<b>10</b>
<b>DESCRIPTION OF PROJECT DATA EXCHANGE STANDARDS.....</b>	<b>10</b>
4.1 STANDARD FOR THE EXCHANGE OF PRODUCT MODEL DATA (STEP) .....	10
4.2 INDUSTRY FOUNDATION CLASSES (IFC) .....	11
4.3 ELECTRONIC DATA INTERCHANGE (EDI) .....	12

4.4 UNITED NATIONS ELECTRONIC DATA INTERCHANGE FOR ADMINISTRATION, COMMERCE AND TRANSPORT (UN/EDIFACT) .....	12
4.5 ORGANISATION FOR THE ADVANCEMENT OF STRUCTURED INFORMATION STANDARDS (OASIS) .....	13
4.5.1 <i>Electronic Business Extensible Markup Language (ebXML)</i> .....	13
4.6 INTERNATIONAL ORGANISATION FOR STANDARDISATION (ISO) .....	16
4.7 ARCHITECTURAL, ENGINEERING AND CONSTRUCTION EXTENSIBLE MARKUP LANGUAGE (AECXML) .....	16
4.8 SUMMARY OF DATA EXCHANGE STANDARDS.....	18
4.9 SUITABILITY OF STANDARDS FOR PROJECT MANAGEMENT IN THE SA GOVERNMENT.....	19
<b>CHAPTER 5.....</b>	<b>20</b>
<b>DESCRIPTION OF SELECTED INFORMATION SYSTEM TECHNOLOGIES FOR PROJECT MANAGEMENT IN GOVERNMENT ....</b>	<b>20</b>
5.1 COMPUTER PROGRAMMING LANGUAGES .....	20
5.1.1 <i>Java</i> .....	21
5.1.2 <i>C++</i> .....	22
5.1.3 <i>C#</i> .....	23
5.1.4 <i>Visual Basic</i> .....	23
5.1.5 <i>SQL</i> .....	24
5.1.6 <i>Summary of Computer Programming Languages</i> .....	24
5.2 MARKUP LANGUAGES AND RELATED TECHNOLOGIES .....	25
5.2.1 <i>Hypertext Markup Language (HTML)</i> .....	25
5.2.2 <i>Extensible Markup Language (XML)</i> .....	25
5.2.3 <i>Document Type Definition (DTD)</i> .....	27
5.2.4 <i>Extensible Stylesheet Language (XSL)</i> .....	28
5.2.5 <i>XML Schema Definition (XSD)</i> .....	28
5.2.6 <i>XSL Transformation (XSLT)</i> .....	29
5.2.7 <i>XML Path Language (XPath)</i> .....	30
5.2.8 <i>Summary of Markup Languages</i> .....	32
5.2.8.1 <i>Applicability to project data processing</i> .....	32
5.3 COMMUNICATION TECHNOLOGIES .....	33

5.3.1 Simple Object Access Protocol (SOAP) .....	33
5.3.2 Remote Method Invocation (RMI).....	34
5.3.3 Remote Procedure Calls (RPC) .....	36
5.3.4 Component Object Model (COM) .....	37
5.3.5 Distributed Component Object Model (DCOM).....	38
5.3.6 Java™ 2 Platform, Enterprise Edition (J2EE).....	39
5.3.7 Microsoft .NET.....	39
5.3.8 Hypertext Transfer Protocol (HTTP) .....	41
5.3.9 Summary of Communication Technologies .....	43
5.4 INTERNET TECHNOLOGIES.....	44
5.4.1 Servers .....	44
5.4.1.2 Apache Tomcat server .....	44
5.4.2 Servlets.....	45
5.2.5 Java Server Pages (JSP).....	46
5.5 DATABASE TECHNOLOGIES .....	47
5.5.1 Database types .....	47
5.5.1.1 Flat-file Databases .....	47
5.5.1.2 Relational Databases.....	48
5.5.1.3 Object orientated database .....	49
5.5.1.4 Summary of database types.....	50
5.5.2 Database processing tools.....	51
5.5.2.1 Java Database Connectivity (JDBC).....	51
5.5.2.2 Open Database Connectivity (ODBC).....	52
5.5.2.3 Summary of database processing tools .....	53
5.6 DOCUMENT MANIPULATION TECHNIQUES AND TOOLS .....	54
5.6.1 Trees .....	54
5.6.2 Document Object Model (DOM).....	55
5.6.3 Java Document Object Model (JDOM).....	56
5.6.4 Simple API for XML (SAX) .....	56
5.6.5 Parsers.....	57

<b>CHAPTER 6.....</b>	<b>58</b>
<b>PROJECT INFORMATION MANAGEMENT.....</b>	<b>58</b>
6.1 PROJECT MANAGEMENT DOCUMENTS.....	58
6.1.1 <i>Application Form</i> .....	58
6.1.2 <i>Business Plan</i> .....	59
6.1.3 <i>Tender Document</i> .....	59
6.1.4 <i>Bill of Quantities</i> .....	60
6.1.5 <i>Contractors Appointment Letter</i> .....	62
6.1.6 <i>Payment Certificate</i> .....	62
6.1.7 <i>Tax Invoice</i> .....	63
6.1.8 <i>Reports</i> .....	64
6.1.9 <i>Variation Orders</i> .....	64
6.1.10 <i>Certificate of Practical Completion</i> .....	65
6.1.11 <i>Form of surety release</i> .....	66
6.1.12 <i>Final certificate</i> .....	67
<b>CHAPTER 7.....</b>	<b>68</b>
<b>PROJECT INFORMATION FLOW.....</b>	<b>68</b>
7.1 PROJECT PROCEDURES .....	68
7.2 PROJECT INFORMATION FLOW IN THE CONSOLIDATED MUNICIPAL INFRASTRUCTURE PROGRAMME .....	70
7.3 PROJECT INFORMATION FLOW DURING CONSTRUCTION.....	72
<b>CHAPTER 8.....</b>	<b>73</b>
<b>ACTIONIT ORGANISATION AND PROJECT INFORMATION MODEL DESCRIPTION.....</b>	<b>73</b>
8.1 ACTIONIT STRUCTURE .....	73
8.2 PROJECTS IN GOVERNMENT MODEL (FIRST VERSION).....	74
8.2.1 <i>System Model</i> .....	75
8.2.2 <i>Process Model</i> .....	76
8.2.3 <i>Enterprise Data Model</i> .....	77



<b>CHAPTER 9.....</b>	<b>80</b>
<b>DESCRIPTION OF IMPLEMENTED APPLICATIONS AND SYSTEM.....</b>	<b>80</b>
9.1 USER REQUIREMENTS .....	80
9.2 APPLICATION AND SYSTEM REQUIREMENTS.....	81
9.2.1 <i>Information</i> .....	81
9.2.2 <i>Time Efficiency</i> .....	81
9.2.3 <i>Data storage</i> .....	81
9.2.4 <i>Aggregation</i> .....	82
9.2.5 <i>Reports</i> .....	82
9.3 APPLICATIONS AND SYSTEM DESIGNS.....	82
9.4 WEB APPLICATION.....	85
9.4.1 <i>Components and Structure</i> .....	85
9.4.1.1 Web pages .....	86
9.4.1.2 Server .....	86
9.4.1.3 Servlets.....	87
9.4.1.4 Database .....	89
9.4.2 <i>Application Functionality</i> .....	92
9.4.2.1 Online Data Entry .....	92
9.4.2.2 Logging of events .....	92
9.4.2.3 Data storage.....	93
9.4.2.4 User feedback.....	93
9.5 XML DOCUMENT CREATION AND AGGREGATION APPLICATION .....	94
9.5.1 <i>Components and Structure</i> .....	94
9.5.1.1 AggregateGui class.....	95
9.5.1.2 TestTreeBuilder class.....	96
9.5.1.3 Aggregate class.....	96
9.5.1.3.3 Server and Client side Aggregation .....	98
9.5.1.4 WriteFile class.....	99
9.5.2 <i>Application Functionality</i> .....	99
9.5.2.1 XML Document creation .....	99
9.5.2.2 Aggregation .....	99
9.5.2.3 Graphical User Interface.....	99

9.6 DATA PROCESSING AND REPORT GENERATION SYSTEM .....	101
9.6.1 <i>Components, Structure and Functionality</i> .....	101
<b>CHAPTER 10.....</b>	<b>105</b>
<b>EXAMPLES OF IMPLEMENTED SYSTEMS AND APPLICATIONS.....</b>	<b>105</b>
10.1 WEB APPLICATION EXAMPLE .....	105
10.2 PROJECT AGGREGATION EXAMPLE .....	108
10.3 DATA PROCESSING AND REPORT GENERATION EXAMPLE .....	112
<b>11. CONCLUSIONS, SUMMARY AND RECOMMENDATIONS .....</b>	<b>120</b>
<b>12. REFERENCES.....</b>	<b>123</b>
<b>APPENDIX A.....</b>	<b>127</b>
<b>COOKTOP2.200 DESCRIPTION AND FUNCTIONALITY .....</b>	<b>127</b>
<b>APPENDIX B.....</b>	<b>133</b>
<b>DB2XML 1.4 DESCRIPTION AND FUNCTIONALITY.....</b>	<b>133</b>
<b>APPENDIX C.....</b>	<b>135</b>
<b>PROCEDURES FOR INSTALLATION OF IMPLEMENTED APPLICATIONS AND SYSTEM .....</b>	<b>135</b>
1. WEB APPLICATION .....	135
1.1 <i>Web pages</i> .....	135
1.2 <i>Server</i> .....	136
1.3 <i>Servlets</i> .....	136
1.4 <i>Database</i> .....	136
2. AGGREGATION APPLICATION .....	138
3. DATA PROCESSING AND REPORT GENERATION SYSTEM .....	140
3.1 <i>Java application</i> .....	140
3.2 <i>XML application</i> .....	140

**APPENDIX D..... 143**

**LIST OF ABBREVIATIONS AND ACRONYMS ..... 143**

**APPENDIX E..... 145**

**REFERENCE TO DATA CD CONTENT..... 145**

## List of Figures

Figure 1 CMIP Management Structure .....	7
Figure 2 ebXML Business Collaboration Process .....	15
Figure 3 Programming language generations.....	21
Figure 4 Example of an XML file .....	26
Figure 5 XSLT Processor functioning .....	30
Figure 6 SOAP Architecture overview .....	34
Figure 7 RMI Architecture .....	35
Figure 8 Remote Procedure Calls .....	36
Figure 9 Client using COM object through an interface pointer .....	38
Figure 10 .NET and XML web services .....	41
Figure 11 HTTP Operation.....	42
Figure 12 Client - Server - servlet relation.....	45
Figure 13 Document Tree structure .....	55
Figure 14 Project procedures and documents.....	69
Figure 15 Project information flow in Government .....	71
Figure 16 Government Structure .....	73
Figure 17 The ActionIT Enterprise Data Model .....	78
Figure 18 Example of accounts for the Enterprise Data Model .....	79
Figure 19 System diagram.....	83
Figure 20 Application Structure .....	85
Figure 21 <i>ApplicationTables</i> table.....	89
Figure 22 <i>xxxxApplicationData</i> table.....	90
Figure 23 <i>Passwords</i> table .....	90
Figure 24 <i>Mapping2</i> table .....	91
Figure 25 <i>xxxxAcountrsRegister</i> table .....	91
Figure 26 Aggregation application classes and data flow.....	94
Figure 27 AggregateGui graphical user interface.....	95
Figure 28 Project Aggregation .....	97
Figure 29 Aggregation of multiple XML documents .....	98
Figure 30 Data Processing.....	102

---

Figure 31 Website homepage .....	105
Figure 32 Application form provided as a web page .....	106
Figure 33 Response after successfully submitting data .....	106
Figure 34 Table created and stored in the database for a given municipality .....	107
Figure 35 Project data table 1 .....	108
Figure 36 Project data table 2 .....	108
Figure 37 Aggregated project XML document .....	111
Figure 38 Extract from the Bill of Quantities.....	112
Figure 39 Bill of Quantities imported into the Access database .....	113
Figure 40 Bill of Quantities converted to XML .....	115
Figure 41 XSL Stylesheet .....	117
Figure 42 XSLT Sample Report in HTML format.....	118
Figure 43 Sample Report output of Java Application.....	119
Figure 44 Example of proposed referencing of elements .....	122

## List of Tables

Table 1 Comparison of Standards.....	18
Table 2 Suitability of standards for project management .....	19
Table 3 Summary of Computer Programming Languages .....	24
Table 4 Markup Language Summary.....	32
Table 5 Communication Technologies .....	43
Table 6 Database type comparison .....	50
Table 7 Database processing tools.....	53

# CHAPTER 1

## Introduction

Project information management has always been an integral part of any project, often critical to the survival of a project. People have always been looking for better, faster and cheaper ways for managing project data and, where these systems were implemented, a remarkable difference in productivity and costs were noticeable. The importance of good management is well illustrated in the following extract from the Financial Mail, October 11, 2002.

*“SA ranks bottom of seven countries surveyed for productivity levels during 2002, and is the only one whose productive time is below 50%. The fault is overwhelmingly management’s, say Proudfoot. Poor systems, including lack of planning and controls, account for 46% of the wastage, and inadequate management another 33%. Lack of skills among the workforce is responsible for only 9%, and the rest is split between IT-related problems, low morale and ineffective communication.”*

In the light of the aforementioned, this thesis presents a discussion on proposed systems for more effective and efficient management of project data by the South African Government. To date, information management has been a slow and expensive process. The thesis starts off with a description of the Consolidated Municipal Infrastructure Programme (CMIP) implemented in the South African Government and then continues to shortly discuss available technologies and how these can be utilised for project management in the public sector. It then continues to describe project management forms, the flow of project data and the ActionIt (a government organisation developing standards for use in the SA Government) project data models. Applications and a system is then developed, conforming to the ActionIt standards, using CMIP as example and making use of Internet technologies. Finally a few examples are discussed to better illustrate the work. The main issues discussed include three components; the faster and more efficient gathering of project data; the aggregation of project data and the use of the Extensible Markup Language (XML)

and related technologies for the generation of project reports. The last component mentioned is of particular interest as it utilises relatively new technologies, which are becoming more and more popular, for its data management capabilities.

The developed system and applications discussed at the end of this thesis are limited in respect that the above mentioned three components are not yet linked, necessitating the manual formatting of data to be used as input for the second and third components. These three components can be linked to form a fully operational system.



## CHAPTER 2

### Thesis Objectives

The objectives of this thesis were to:

1. Develop an understanding of project management in the South African Government and public sector.
2. Describe the ActionIT (a government organisation developing standards for use in the SA Government) data exchange concepts and to develop pilot applications conforming to the ActionIT specifications to:
  - a) put a platform in place for online data entry.
  - b) store the gathered data.
  - c) extract and convert the data to the Extensible Markup Language (XML) format.
  - d) transform the data to comply with the ActionIT standard.
  - e) generate reports from this data.
  - f) contribute to the validation of the ActionIT models for the civil engineering industry.
3. Implement and demonstrate the above system.
4. Study suitable technologies and system implementations that were relevant to points one and two.

## CHAPTER 3

# **An Project Management System in the South African Government**

This chapter deals with a project management system currently used by the South African Government to manage and administrate the flow of project information. This research was important as it provided the knowledge needed to create and manage an online project data entry system. The *Consolidated Municipal Infrastructure (Government) Programme* and the *Per-Form Developer* computer programme were researched and will be discussed below.

### **3.1 Consolidated Municipal Infrastructure Programme (CMIP)**

The Consolidated Municipal Infrastructure Programme (CMIP) is a South African Government initiative by the Department of Provincial Local Government (DPLG). It is responsible for receiving funds from the government and distributing these funds for use in local authorities. CMIP is aimed at providing basic levels of services to low-income households. It also aims at contributing to other government strategic and intervention policy objectives, which include:

- small, medium and micro-sized enterprise (SMME) development.
- utilisation and empowerment of affirmative business enterprises.

The CMIP primary objectives are to provide funding for municipal infrastructures (such as water, sanitation, roads, solid waste, community lighting, storm water, community facilities and training), to support the government housing programme and to provide funding to municipalities. This is to minimise backlogs by providing basic levels of infrastructure services to low-income households from the year 1997 to 2007.

### **3.1.1 CMIP Management structure:**

CMIP is managed on three levels: national, provincial and municipal. Below is a description of the different levels.

#### **National:**

At national level a municipal infrastructure task team is appointed to ensure a coordinated approach to CMIP (See Figure 1). This includes different national departments as well as the Development Bank of Southern Africa. The person responsible for CMIP on national level is the Minister of Provincial Affairs and Constitutional Development. The Department of Provincial Local Government (DPLG) is the national government agent responsible for managing CMIP and convenes the municipal infrastructure task team. A national programme manager is appointed by the DPLG and heads a national municipal infrastructure task team.

Coordination and communication between the national and provincial levels takes place through MinMec. MinMec is a forum, which comprises of the national Minister of Provincial Affairs and Constitutional Development, as well as the Members of the Executive Council (MEC) for the local governments of the provinces.

#### **Provincial:**

At the provincial level the MEC for local government is responsible for CMIP (See Figure 1). The department of the MEC is responsible for leading a municipal infrastructure task team. The job of the team is to ensure coordination between departments at the provincial level. Each provincial department in the local government is responsible for appointing a provincial programme manager to head the provincial teams. The purpose of these teams is to support and assist municipalities as requested.

#### **Municipal:**

On municipal level municipalities are responsible for:

- establishing project teams for each approved CMIP project.

- preparing all applications for funds as well as documentation of projects, which include technical reports, business plans, contracts and monitoring of projects.
- reporting any irregularities to the provincial programme manager.
- taking over a completed project as an asset of the municipality.

[*'The Consolidated Municipal Infrastructure Program Handbook'*, available at: [www.local.gov.za/](http://www.local.gov.za/)]

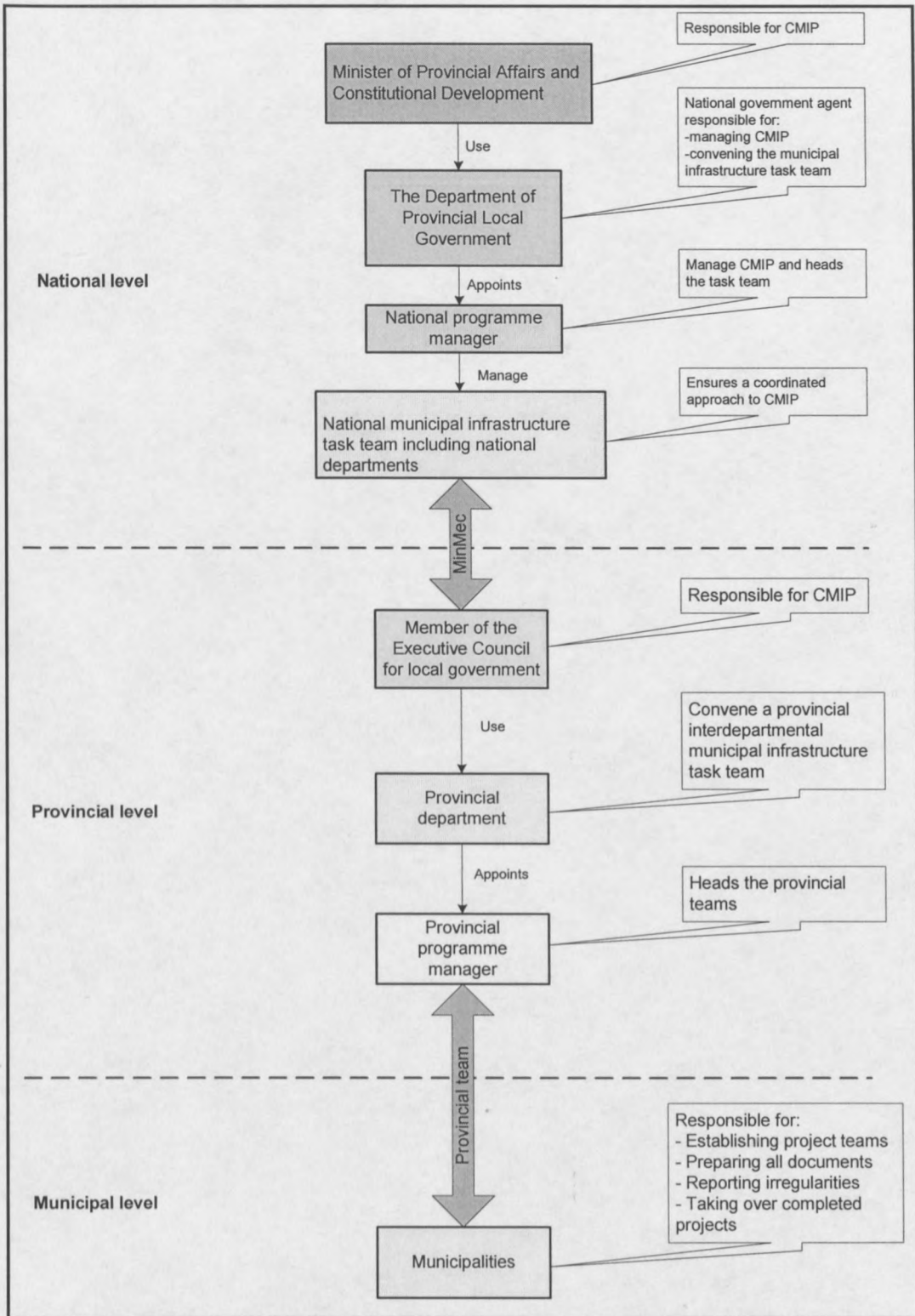


Figure 1 CMIP Management Structure

### **3.1.2 CMIP Functioning**

To accomplish the CMIP goals, CMIP created a system whereby municipalities apply for project funding by filling out prescribed forms and submitting these to CMIP. Each application is then prioritised according to prescribed guidelines. These guidelines include:

- local-community priorities.
- spatial efficiency.
- job-creation potential.
- the implementation time of the project.
- counter-funding opportunities.

After prioritisation, the application is either approved or sent back for modification. On approval of a project application, the municipality is requested to submit a business plan (see section 6.1.2 Business Plan) for the project. If the business plan is approved by CMIP, the project is sent to the provincial cabinet that will, on approval of the project, endorse it. After this the endorsement is sent to the national government where the funding is approved and distributed accordingly (see Figure 15).

The forms to be completed are available on the CMIP website, [www.cmip.co.za](http://www.cmip.co.za), which must be printed, completed and sent to CMIP. After the forms have been received, the information is entered into the PER-FORM Developer computer programme.

### **3.2 PER-FORM Developer**

The PER-FORM Developer management system is a system developed for the organisation and management of project data. The system was developed by *Inform Systems* and aims at improving the conventional project tracking and management consultant approaches to programme management. Currently the system facilitates a web interface with the following capabilities: [taken from [www.in-forms.co.za](http://www.in-forms.co.za)]

- Forms captured on the Web are retained as separate forms and can be imported into PER-FORM Developer once their status warrants it.
- Forms can be forwarded to a number of people for review before finally being submitted.
- Automatic e-mail notification of submission and allocation of forms can be made.
- Forms can be printed using a version that is optimised for printing.
- Forms are pre-filled with any available data from PER-FORM Developer.
- Detailed validation is done on the form, prior to allowing the user to submit it.
- The form can be saved temporarily prior to being submitted.
- Once a user has submitted a form, the form can no longer be edited.
- Users can register themselves on the Web but only have access to projects allocated to them.

The PER-FORM Developer system is currently used by CMIP as a project data management facility. [[www.in-forms.co.za](http://www.in-forms.co.za)]

## CHAPTER 4

### **Description of Project Data Exchange Standards**

Standards are agreed upon technical specifications or precise criteria to be used in a consistent manner as rules and guidelines for the consistent creation of processes, services and physical products. This ensures the usability and interoperability over national and international borders.

Standards are administrated by standards organisations, which can be governmental or non-governmental organisations. These organisations are responsible for the creation of standards and each have their own procedures and guidelines as how to create and maintain standards. The procedures for creating standards usually consist of a proposal stage, an introductory stage for creating an introductory standard, reviewing and recommendations of the standard by technical committees, release for community reviewing, approval of the standard and final publication.

Included in the standard descriptions below are selected paraphrased paragraphs taken from the indicated references, with quotes showing direct quotations.

#### **4.1 Standard for the Exchange of Product model data (STEP)**

The Standard for the Exchange of Product model data (STEP) is an international effort under the International Standards Organisation, contributed to by national standards organisations such as the South African Bureau of Standards, the British Standards and the American National Standards Institute. Its aim is to produce high-level standards for exchanging product information that support technical information exchange and communication within industries. "STEP data models are rich, structured and object-based models of industrial products. They are generally designed to provide fairly complete, cross-application descriptions of product data and the modelling rules and guidelines for defining them are extensive and formal." As a result, STEP models are complex to develop and use, but they provide the



possibility of enabling widespread exchange of complex technical information among various computing applications throughout industries. Currently STEP models are incorporated into commercial software focusing on the nuclear, automotive and shipping industry.

[[www.civil.ubc.ca](http://www.civil.ubc.ca)]

## **4.2 Industry Foundation Classes (IFC)**

The Industry Foundation Classes (IFC) is an initiative under the International Alliance for Interoperability (IOI). It is contributed to by the Architectural, Engineering and Construction industries as well as by '*facility management*' software vendors. The IFC focuses on the building industry and is widely used in Computer Aided Drawing (CAD). "It provides a means to encode and store information for an entire project in a model that can be shared among diverse project participants. The information contained in such a model can include project management information such as estimating and scheduling data as well as physical construction data."

IFC enables the passing of a complete, thorough and accurate data model from one computer application to another without any loss of information.

The classes of the IFC describe the building model, its components and the relationships between them. The project model for a project constitutes an object-oriented database of the information shared among project participants and continues to grow as the project goes through design, construction and operation. The information in the database consists of classes, which represent the parts of buildings or elements of the process and includes the relevant information about these parts.

The procedures for developing these standards are described in the IFC Modelling Guide and the IFC Specification Development Guide which can be found on the IFC website.

[<http://itcon.org/1999/2/> , [http://cig.bre.co.uk/iai\\_uk/iai/page4.html](http://cig.bre.co.uk/iai_uk/iai/page4.html)]

### **4.3 Electronic Data Interchange (EDI)**

Electronic data interchange started as an initiative of the transport industry in the seventies to find a way to overcome the inefficiency and high cost associated with paperwork. From this developed the international standard for Electronic Data Interchange under the auspices of the United Nations Economic Commission for Europe (UN/ECE) and the International Standards Organisation. EDI aims at being a structured, efficient way of conveying messages between businesses while taking advantage of modern information technology. It makes the formatting of data, according to an agreed standard, possible by specifying these standards, thus facilitating the electronic transfer from one computer system to another. The goal is to have the computer systems of two business partners exchanging business information without human intervention.

As EDI was developed to provide for the need of individual companies, the need for international standards soon became necessary for trading between international companies and different industries. This led to the founding of the United Nations Electronic Data Interchange For Administration, Commerce and Transport (UN/EDIFACT). [<http://edocs.bea.com/wli/docs70/edi/>, [www.uncec.org](http://www.uncec.org)]

### **4.4 United Nations Electronic Data Interchange For Administration, Commerce and Transport (UN/EDIFACT)**

UN/EDIFACT evolved from EDI to provide a means for cross industry and international trading by providing international standards. It was founded in 1986 by the United Nations Economic Commission for Europe (UN/ECE) and the International Standards Organisation and aims at providing a single international EDI standard flexible enough to meet the needs of government and private industry. Currently EDIFACT is to be superseded by ebXML (See [4.5.1 Electronic Business Extensible Markup Language \(ebXML\)](#)). [[www.unedifact.com](http://www.unedifact.com), [www.uncec.org](http://www.uncec.org)]

## **4.5 Organisation for the Advancement of Structured Information Standards (OASIS)**

OASIS is an organisation created to promote and encourage the use of structured information standards such as the Extensible Markup Language (XML) and the Standard General Markup Language (SGML). This goal is accomplished by OASIS' initiative to:

- Develop XML and SGML applications such as XML schema/Data Type Definitions (DTD), name spaces and style sheets.
- Develop specifications defining how e-business systems should be built.
- Develop specifications and standards that define how other standards will work together.
- Develop test scenarios and cases that can determine what it means to conform to specific standards.

This is achieved through technical committees governing all technical work. OASIS is an international organisation. "To give all parties equal say in the creation of technical work, it provides a vendor neutral home for the development of technical work as well as a democratic process for deciding if the standard created by the committee is acceptable."

One of the major initiatives by OASIS, in co-operation with the United Nations body for Trade Facilitation and Electronic Business, was the creation of the Electronic Business Extensible Markup Language (ebXML).

[<http://www.oasis-open.org>]

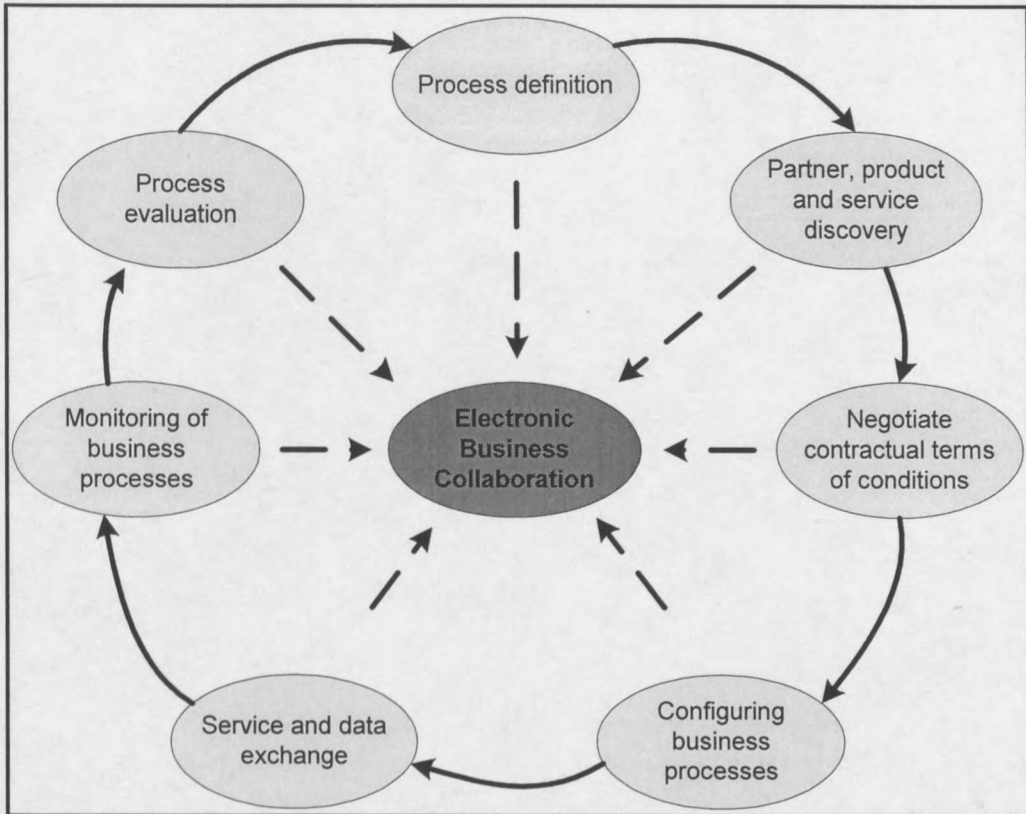
### **4.5.1 Electronic Business Extensible Markup Language (ebXML)**

The Electronic Business XML (ebXML) is an XML-based framework created to enable the use of electronic business information in an interoperable, secure and

consistent manner. Many different parties, regardless of size or location, are enabled to participate in electronic business collaborations with their business partners through the exchange of XML-based business messages (Figure 2).

In order to facilitate these collaborations, businesses need to:

- Determine and define which business processes will be necessary for electronic commerce. These business processes are defined according to a well-known model and described in agreed upon formats.
- Find suitable business partners by evaluating other business' profiles, and determine the services and products offered.
- Negotiate an agreement on contractual terms and conditions for the collaboration, thus determining how data exchange will take place.
- Configure the agreed upon business interfaces and documents needed for the collaboration. This is the implementation of the business process.
- Exchange of services and products following the agreements.
- Monitoring of the business processes for compliance with trading partner agreements and successful execution.
- Evaluate the business process, improving and reconstructing where necessary.



**Figure 2 ebXML Business Collaboration Process**

All of the above requirements are provided for in ebXMLs infrastructure as it makes provision for:

- data communication interoperability.
- a semantic framework for commercial interoperability.
- a mechanism that allows enterprises to find and establish relationships and conduct business with each other. [[www.ebxml.org](http://www.ebxml.org)]

## **4.6 International Organisation for Standardisation (ISO)**

The International Organisation for Standardisation (ISO) is a non-governmental organisation consisting of a federation of standard bodies from countries worldwide and is based in Switzerland. "Its purpose is to promote the development of standards to facilitate the international exchange of products and services as well as to develop collaboration between spheres of intellectual, scientific, technological and economic activity."

The ISO accept a standard to be international when all of its members agree upon a standard. The standards are create by ISO technical committees in a six step process (taken from the ISO website):

- Stage 1: Proposal stage
- Stage 2: Preparatory stage
- Stage 3: Committee stage
- Stage 4: Enquiry stage
- Stage 5: Approval stage
- Stage 6: Publication stage

The different stages are described on the ISO website. [[www.iso.ch](http://www.iso.ch)]

## **4.7 Architectural, Engineering and Construction Extensible Markup Language (aecXML)**

The Architectural, Engineering and Construction Extensible Markup Language (aecXML) was initiated by Bentley Systems, but is currently under the supervision of the International Alliance for Interoperability. The rules and policies for the aecXML

standard are being developed by technical committees, who use these rules to create aecXML schemas.

“aecXML is a data format based on well-defined business cases.” It is designed for the purpose of transferring information specific to the architectural, engineering and construction industries over the Internet, free from human intervention. “It aims to be an organised collection of information packets required for a business transaction. These packets are to be transferred or published via the Internet.” They only contain the data, which is necessary for the transaction at hand and not all of the project data. aecXML also strives to establish standard ways of structuring these information packets so as to enable automated processing of the data.

aecXML further provides a set of keywords and named data attributes, enabling users to employ the same naming logic and grouping. This standardisation enables software to make use of the data without it needing to be interpreted by humans and manually re-entering in each programme’s required form.

[[www.aecxml.org/](http://www.aecxml.org/), [www.iai-na.com/files/aecXML Framework R1.doc](http://www.iai-na.com/files/aecXML_Framework_R1.doc). ]

## 4.8 Summary of Data Exchange Standards

The data exchange standards discussed above are summarised in Table 1. The table compares the different standards by listing the uses of each standard, the industry focus and other attributes as seen in the table column headings.

**Table 1 Comparison of Standards**

Standard	Controlling body	Industry Focus	Participating Organisations	Standards development Procedure/status	Format of Standards	Implementation/ aim
<b>STEP</b>	International Organisation for Standardisation (ISO)	Nuclear, automotive, shipbuilding industry	National Standards Organisations eg. SABS, BS, ANSI	Certified for use.	Express language, modules with technical and industry focus.	Aims at providing high-level standards for the exchange of product data while supporting technical information exchange. Software products implementing STEP commercially available.
<b>IFC</b>	International Alliance for Interoperability (IAI)	Building industry, CAD industry, AutoDesk	All AEC/Facilitating management software vendors.	Conforming to the IFC Model Guide and the IFC Specification Development Guide	Classes describing objects or 'things' with common characteristics, eg. doors or windows.	Enabling the sharing of project model data between different software applications in an electronic form.
<b>ebXML</b>	United Nations body for Trade Facilitation and Electronic Business, OASIS and the Joint Coordination Committee (JCC)	All businesses wanting to utilise business to business communication and exchange of business data.	All OASIS, UNCETF members and approved parties.	Technical Committee Processes	Electronic business XML schemas.	Providing a means for the electronic exchange of business data between different enterprises.
<b>EDIFACT</b>	United Nations/Economic Commission for Europe (UN/ECE) and the International Organisation for Standardisation (ISO)	Governments and private industries wanting to exchange data electronically.	UN/ECE, ISO and their sub-structures, SABS.	To be superseded by ebXML	Rules and guidelines.	Creating a single international EDI standard flexible enough to meet the needs of government and private industry.
<b>aecXML</b>	Initiated by Bentley Systems, currently managed by the IAI	Architectural, engineering and construction industry.	Bentley Systems, IAI and a variety of different companies involved in the aec trade.	aecXML Technical Committee develop rules and policies.	AEC-XML schemas	To be a set of AEC- specific XML schemas to facilitate e-business and communication between participants.



## 4.9 Suitability of Standards for project management in the SA Government

What follows is a table reviewing the suitability of some standards, which are used for project management in the South African Government. Suitability for project management purposes is measured against industry focus, complexity, development cost, and the aim of the standard.

**Table 2 Suitability of standards for project management**

Standard	Industry focus	Aim	Complexity	Development Cost/Time	Comment	Suitable for project management in the South African Government
STEP	Nuclear, automotive, shipbuilding industry	Aims at providing high-level standards for the exchange of product data while supporting technical information exchange. Software products implementing STEP commercially available.	Complex to develop.	High cost, long time.	STEP models are complex, expensive and focus on the description and exchange of product data and not on the exchange of a project's data.	No
IFC	CAD industry, AutoDesk, Building industry.	Enabling the sharing of project model data between different software applications in an electronic form.	Medium	Extensive development needed to obtain a IFC, thus a long development time.	IFC is suitable for project management as it already provides classes for project management data and physical construction data.	Yes
ebXML	All businesses wanting to utilise business to business communication and exchange of business data.	Providing a means for the electronic exchange of business data between different enterprises.	Medium	Takes long to implement.	The principals of ebXML can be used to implement a collaboration between Government and municipalities.	Yes
EDIFACT	Governments and private industries wanting to exchange data electronically.	Creating a single international EDI standard flexible enough to meet the needs of government and private industry.	Medium	Depends on application to which the rules and guidelines is applied.	To be superseded by ebXML.	No
aecXML	Architectural, engineering and construction industry.	To be a set of AEC- specific XML schemas to facilitate e-business and communication between participants.	Medium	Medium cost	aecXML is focused on the exchange of data within the AEC community and not between the AEC community and the Government. It can however be extended to include this.	Limited

## CHAPTER 5

# Description of Selected Information System Technologies for Project Management in Government

Information System Technologies for Project Management include hardware, software and software applications. The software includes different building blocks (other computer programmes, programming languages, markup languages, communication technologies, Internet technologies, database technologies, document manipulation tools) for the software applications, while the software application implements these building blocks to provide functional computer programmes for Project Management.

Project Management requires the processing of data, where data processing is defined as a collection of programmes and technologies to organise and manipulate data. Data processing is achieved by creating entire applications written in one programming language or through combinations of selected Information System Technologies.

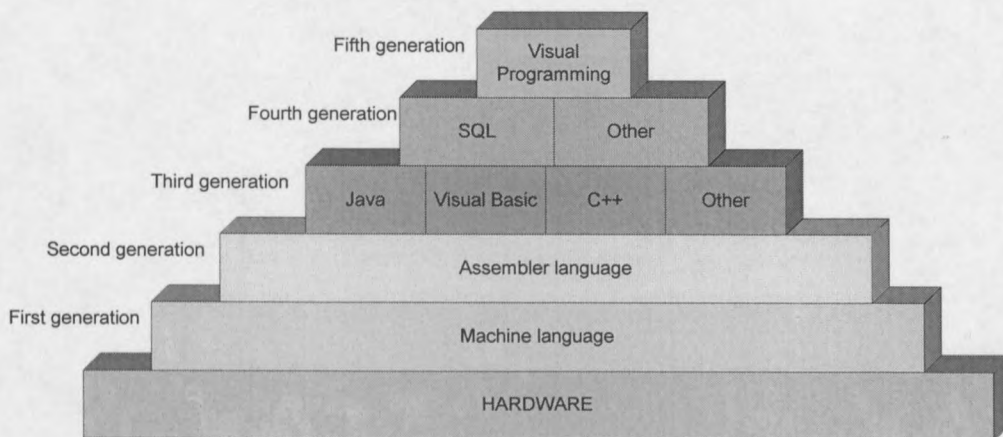
*Figure 3 Programming language generations*

### 5.1 Computer Programming Languages

A computer programming language is a vocabulary and a set of grammatical rules to instruct the computer what to do. Each language defines its own syntax and a set of special keywords. The keywords are words that are understood by the language.

Computer languages can be classified into different generations (Figure 3), usually generations one to five, according to their functionality. Generation one languages, called machine language, are the most basic, a binary instruction of zeros and ones, given to the processor to work on. The second-generation languages are assembler languages. These are similar to machine language, but allow the programmer to use names for numbers, where machine language consists of numbers only. This simplifies programming considerably. Third generation languages are called 'high-

level' programming languages, languages like Java, C, C++ and Visual Basic. These languages use compilers to convert their statements (source code) to machine language. In the case of Java the statements are converted to byte code. Fourth generation languages are the computer languages that are the closest to human language, like the Structured Query Language. Fifth generation languages consist of graphical programming interfaces used to construct third and fourth generation source code, like the Borland JBuilder.



**Figure 3 Programming language generations**

Application programming usually uses third generation languages and the choice of which one to use depends on the functionality of the programme, which machine it is to run on and the skill and personal preference of the programmer.

[[www.wepopedio.com](http://www.wepopedio.com), [www.whatis.com](http://www.whatis.com)]

### 5.1.1 Java

Java is an object orientated, platform independent programming language developed by Sun Microsystems and derived from C and C++, taking all the best properties and ideas of these languages and combining it into Java.

The Java language was published for the first time in 1995 on the Internet and developed from there to become known as ‘*the programming language for the Internet*’. This was due to the fact that Java made network communications easy and provided platform independent code. This meant that code being written on one machine could be compiled to ‘*byte code*’ and without being changed, run on any other machine. As Java was derived from C and C++, it is easily learned by programmers familiar with these languages. It also provides safer, more stable programmes than in C and C++, but in comparison with these languages compromise speed and efficiency for this. Another drawback of Java is that although code being written once can run anywhere, it still needs to be tested on other machines, consuming time.

“Today any Java application can easily be delivered over the Internet, or any network, without operating system or hardware platform compatibility issues.” The Java Technology is currently also being built into next-generation cell phones, television set-top boxes, smart cards and many other consumer and business devices.  
[[www.sun.com](http://www.sun.com)]

### **5.1.2 C++**

C++ is an object orientated language, developed by Sun Microsystems as a fast and efficient language for producing any kind of software, from games to complicated network applications. It is different from Java in that it is not platform independent. C++ code is compiled for a specific platform and after compilation only runs on that specific platform. C++ however, is fast and efficient. “This is due to the fact that programmes run directly on the processor and can use as much performance as needed, greatly improving the executing speed of programmes.” Most operating systems are written in C or C++, leveraging the power of a C++ programme as the functional calls of the operating system are exposed in C++. This means that a C++ programme can tell the operating system exactly what to do when. “A drawback of C++ is that it has no runtime error checking, meaning that if the application tries to access non-existing memory during runtime it will cause the system to stop responding.” It is also a difficult and complex language to learn. It provides

functionality for manipulating the computer's memory and for directly accessing the computer's hardware, contributing to more complex code. [[www.tmdc.org/cplusplus](http://www.tmdc.org/cplusplus)]

### **5.1.3 C#**

C# is an object-oriented language developed by Microsoft. It provides programmers with a fast and efficient way to build a wide range of applications for the Microsoft .NET platform. This platform provides tools and services that extensively use both computing and communications.

"C# also has a high degree of conformity with C and C++, as it was developed to give C and C++ programmers a development tool for rapid programme development without sacrificing the power and control of C and C++." An advantage of C# is that it was designed to shorten the development cycle, helping developers to do more with fewer lines of code and fewer opportunities for error.

C# programmers can leverage an extensive framework for building applications on the Microsoft .NET platform. C# includes built-in support to turn any component, from high-level business objects to system-level applications, into an XML Web service (see 5.3.7 Microsoft .NET). These services can then be invoked over the Internet from any application running on any platform. [[www.msdm.Microsoft.com](http://www.msdm.Microsoft.com)]

### **5.1.4 Visual Basic**

Visual Basic (VB) was developed by Microsoft and released in 1991 as its first visual development tool. It provides extensive support for easily creating the user interface to applications, allowing the user to draw the interface with the mouse and fill in the code from the keyboard. It further provides for the easy access of databases as well as built-in support for accessing the Internet. Since the appearance of VB5 the ability to create ActiveX controls, ActiveX EXEs and ActiveX Dynamic Link Libraries were included into VB. This allows programmers to effortlessly access and manage Internet

works on Microsoft operating systems. Compared to C, C++ and Java, VB is slow and inefficient, using a considerable amount more hard disk space than an equivalent C++ programme would. [[www.visualbasic.com/](http://www.visualbasic.com/)]

### **5.1.5 SQL**

The Structured Query Language (SQL) is a standardised query language used to query multiple different databases and either adds, subtracts or modifies data in the databases. This query language was standardised by the American National Standards Institute (ANSI) for the first time in 1986 and gave applications a common core allowing programmers to easily connect and communicate to different database products. [SQL A Beginner's Guide, Forrest Houlette, Ph. D., McGraw-Hill, 2001]

### **5.1.6 Summary of Computer Programming Languages**

Following is a table summarising the programming languages discussed above. The table provides criteria to assist in the selection of a language that will suite a specific application the best.

**Table 3 Summary of Computer Programming Languages**

applications. VB is however not platform independent and limits portability as it only works on Microsoft operating systems. Compared to C, C++ and Java, VB is slow and inefficient, using a considerable amount more hard disk space than an equivalent C++ programme would. [[www.visualbasic.com/](http://www.visualbasic.com/)]

### 5.1.5 SQL

The Structured Query Language (SQL) is a standardised query language used to query multiple different databases and either adds, subtracts or modifies data in the databases. This query language was standardised by the American National Standards Institute (ANSI) for the first time in 1986 and gave applications a common core allowing programmers to easily connect and communicate to different database products.

[SQL A Beginner's Guide, Forrest Houlette, Ph. D., McGraw-Hill, 2001]

### 5.1.6 Summary of Computer Programming Languages

Following is a table summarising the programming languages discussed above. The table provides criteria to assist in the selection of a language that will suite a specific application the best.

**Table 3 Summary of Computer Programming Languages**

Language	Controlling body	Platform Independent	Object Orientated	Generation/level	Used for	Advantages	Disadvantages
Java	Sun Microsystems	Yes	Yes	High level language. Third generation.	Producing any kind of software, especially internet applications.	Platform independent code, write once run everywhere. Simplify internet programming. Derived from C and C++. Safer and more stable programmes can be created with Java than with C and C++.	Slower and less efficient than C and C++. Difficult to learn.
C++	Sun Microsystems	No	Yes	High level language. Third generation, also leaning to second generation.	Producing any kind of software.	Fast and efficient. Most operating systems are written in C++. Code is very brief. C code can easily be incorporated in C++ code.	Not platform independent. C++ has no runtime error checking. Difficult to learn and read.
C#	Microsoft	No	Yes	High level language. Third generation.	Building applications for the Microsoft.NET platform.	High degree of fidelity with C and C++. Rapid program development. Built in support for creating XML web services.	Not platform independent. Slower than C++. Limited to Microsoft operating systems.
Visual Basic	Microsoft	No	No	High level language. Third generation.	Producing any kind of software.	Extensive support for creating graphical user interfaces. Easy database access. Easy to learn. Fast prototype creation.	Not platform independent. Slow and inefficient. Limited portability as it only works on Microsoft platforms.

## **5.2 Markup Languages and related technologies**

Markup is a sequence of characters or other symbols inserted into a text or word processing file and is often referred to as tags. It can be used to describe the logical structure of a document or how a document should be displayed and printed. Markup languages are a vocabulary of tags and a set of grammatical rules. Technologies relating to markup languages are technologies created to manipulate and use these languages.

### **5.2.1 Hypertext Markup Language (HTML)**

The Hypertext Markup Language is the language most commonly used on the World Wide Web for creating and laying out of documents. HTML is a non-proprietary format based upon the Standard General Markup Language (SGML) and can be created and processed by a wide range of tools. These vary from simple plain text editors, where you type the code from scratch, to sophisticated “What you see is what you get” (WYSIWYG) authoring tools. HTML uses tags such as `<h1>` and `</h1>` to structure text into headings, paragraphs, lists, hypertext links and so forth.

### **5.2.2 Extensible Markup Language (XML)**

XML is the abbreviation for the Extensible Markup Language and is a World Wide Web Consortium (W3C) -endorsed standard for document markup.

“XML enables the open exchange of information, including structured data elements and was originally designed for large-scale document transportation and computer based processing. It provides a structured data format suitable for direct application to application interfaces, as well as displaying the data in a web browser.”

The design goals for XML, as defined by the W3C, are [taken from [www.w3c.org/](http://www.w3c.org/)]:



1. XML shall be easy to use over the Internet.
2. XML shall support a wide variety of applications.
3. XML shall be compatible with SGML.
4. It shall be easy to write programmes which process XML documents.
5. The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
6. XML documents should be human-legible and reasonably clear.
7. The XML design should be prepared quickly.
8. The design of XML shall be formal and concise.
9. XML documents shall be easy to create.
10. Terseness in XML markup is of minimal importance.

Conforming to these goals XML provides easy human readable tags (Figure 4) for describing data as well as storing all data as strings surrounded by these tags, thus readable data. A XML file contains elements which are specific units of data and its markup, for example `<name>Asla</name>`. An element can be nested within other elements and can contain attributes.

```
<?xml version="1.0"?>
<Project name= "road">
  <contractor>
    <name>Asla</name>
    <refno>001232</refno>
    <date>15/11/2002</date>
  </contractor>
</Project>
```

**Figure 4 Example of an XML file**

XML does not specify a predefined tag set or prescribed language grammar, where the *tag set* of a language defines the tags that have meaning to a language parser and

the language *grammar* defining the way in which the tags should be used. By not specifying either of these, XML accomplishes its goal of being completely extensible. XML specifies a set of grammatical rules of what an element should look like. This includes rules of how tags should delimit elements, where attributes should be placed and how the Document Type Declaration (DTD) or XML Schema Definition (XSD) should be created.

Currently XML is a widely used Internet data language:

- Allowing data to be freely exchanged between different platforms and applications.
- Giving data a longer life span as it can be used by different applications.
- Facilitating local and specific data by using XMLs extensibility.
- Establishing standards for the exchange of data between two parties.

### **5.2.3 Document Type Definition (DTD)**

Document Type Definitions forms part of the XML specification and is used to describe how the data of a XML document must be formatted. A DTD is made up of a document type declaration, the head of a XML document, and can contain markup constraints and/or refer to an external document for markup constraints. These constraints specify how the data in the XML document should be constructed. This is important as it enables other applications or parties to understand what the tags for this document mean. DTD can specify how elements must be constructed, which elements and attributes are allowed, the range of attribute values, the amount of times elements or attributes may appear and other more complicated constraints. [Java and XML, Brett McLaughlin, 2000]

### **5.2.4 Extensible Stylesheet Language (XSL)**

The Extensible Stylesheet Language was developed by the World Wide Web Consortium (W3C) and is used to create stylesheets for describing how a XML document should be displayed. The focus is thus on the translation and transformation of XML data from one XML format into another. For this reason XSL is an excellent choice when a XML document needs to be mapped between different formats, for instance if it is available in HTML, but also needs to be displayed as Portable Data Format or Postscript form.

One of the ways in which XSL performs translations is by using formatting objects. These are specific tags, which can be replaced by content specific to the document type the data is being translated to. This implies, for example, that it can be specified that all <hallo> tags in a document must be replaced with <welcome> tags, or any other operation done to the specific tag and its content when it is encountered in the document.

[[www.whatis.techtarget.com/XSL/](http://www.whatis.techtarget.com/XSL/), Java and XML, Brett McLaughlin, 2000]

### **5.2.5 XML Schema Definition (XSD)**

The XML Schema Definition is a World Wide Web Consortium (W3C) recommendation used for specifying how elements in a XML document should be described. XML documents can then be verified against these XSDs to determine if all the content in the document adheres to the element descriptions in which they occur. The schema referred to in the title defines the interrelationship between the attributes and elements of a XML document. It is in itself a XML document that uses specialised XML elements and attributes that describe the changes that must be made to a document. “When creating a XSD for a XML document the structure of the document will be analysed and all structural elements will be defined as they are encountered.”

Currently the XSD standard consists of two documents as well as a tutorial, listed below:

- XML Schemas Part 1: Structures
- XML Schemas Part 2: Data types
- XML Schemas Part 0: Primer

The advantage of XSD above earlier schema type definitions, such as DTD, is that XSD are written in XML, thus not requiring intermediate processing. XSD further includes self-documentation, automatic schema creation and the ability to be queried through XML Transformations. [[www.whatis.techtarget.com/sDefinitions](http://www.whatis.techtarget.com/sDefinitions), [www.vbxml.com/xml/articles/xsd/kurt\\_schema2.asp](http://www.vbxml.com/xml/articles/xsd/kurt_schema2.asp), [www.topxml.com/xsl](http://www.topxml.com/xsl)]

### **5.2.6 XSL Transformation (XSLT)**

XSLT is part of the XSL specification and is used for changing the structure of a XML document. This is done by reading the input XML document into a source tree and then building a result tree from the source tree. This process is handled by a XSLT processor, taking a XML document and a stylesheet as input and returning a XML document with a changed structure as output (see Figure 5). The changes to the document are specified in the stylesheet. XSLT code is also referred to as a stylesheet and can be used in combination with a XSL stylesheet or can be used on its own.

XSLT provides functionality for matching patterns within the original XML document and applying changes to this data. It also provides syntax for common operators, pattern matching and more, all intended to make the transformation of a XML document to another format easier. [[www.whatis.techtarget.com/sDefinitions](http://www.whatis.techtarget.com/sDefinitions), Java and XML, Brett McLaughlin, 2000]

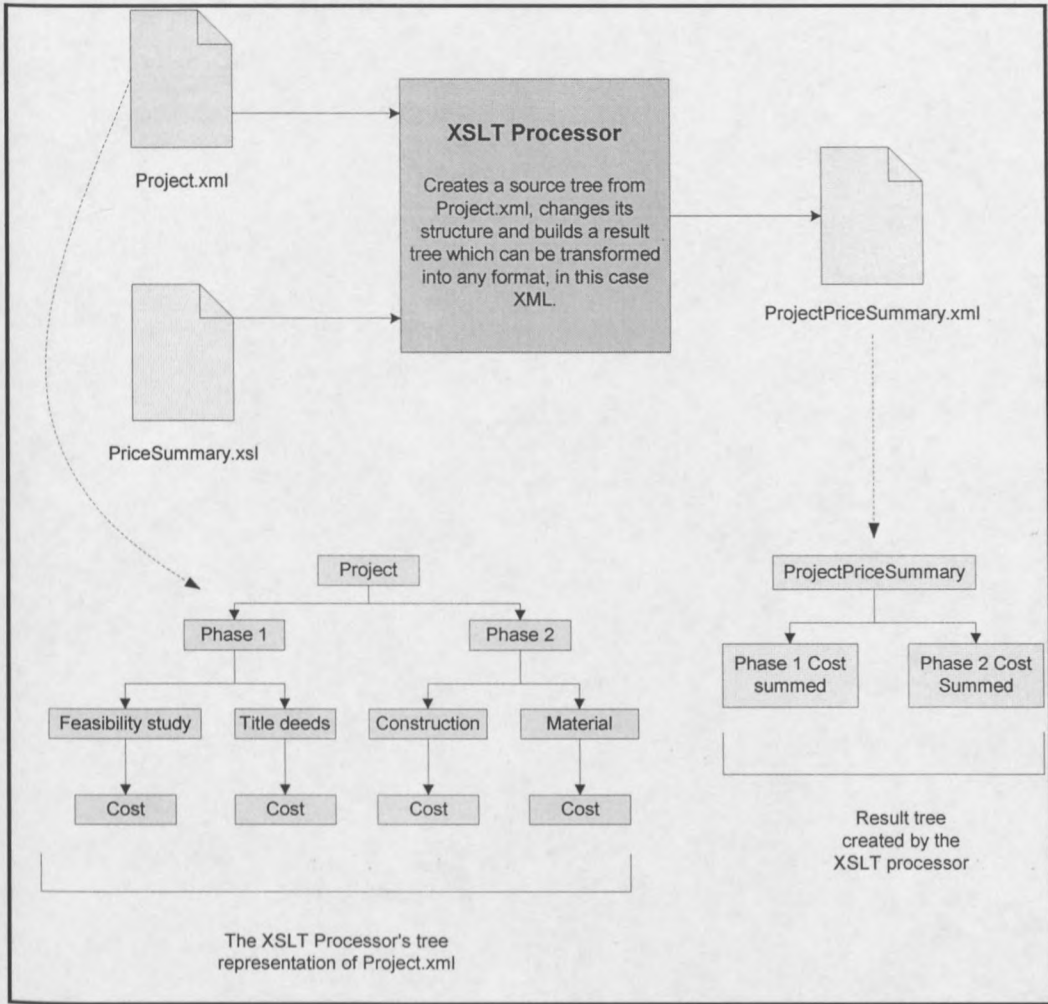


Figure 5 XSLT Processor functioning

### 5.2.7 XML Path Language (XPath)

The XML Path Language provides the functionality needed to refer to all the different element and attribute names and values in a XML document. This is important as a XML document can be complex and finding specific elements and attributes may be difficult. XPath provides the syntax necessary to locate any element or attribute in a XML document and it performs this operation by specifying the path to any other node relative to the current node. XPath also provides references to nodes relative to

the root of the document, providing for cases where an element outside of the current element's scope must be referenced. [<http://www.w3.org/TR/1999/REC-xpath-19991116>]

## 5.2.8 Summary of Markup Languages

Following is a table summarising the markup languages discussed above. The table provides criteria to assist in the selection of a language that will suite a specific application the best.

**Table 4 Markup Language Summary**

Language	Controlling body	Platform Independent	Object Orientated	Generation/level	Used for	Advantages	Disadvantages
HTML	W3C	Yes	No	-	Creating and laying out of documents.	Most common language used on the Internet. Can be created and processed by a wide range of tools.	Not extensible. Does not support deep data structures needed to represent databases. Does not support validation for applications to check data for structural validity on importation.
XML	W3C	Yes	No	-	Open exchange of data between different applications.	Self defined tags can be used. Human readable. Support deep data structures. Support complex nesting of elements. Extensible. Support validation. Supporting technologies include XSD,DTD,XSLT,XPATH	Not backward-compatible with existing HTML documents.

### 5.2.8.1 Applicability to project data processing

The above technologies and languages are all suitable for project data processing as it all provides mechanisms for data manipulation, either on its own or combined with other technologies. Some programming languages however will work better than others, depending on what is to be achieved and the language to be used must be decided upon this criteria.

## **5.3 Communication Technologies**

Communication is the transmission of data from one party/computer/device to another. Communication Technologies include communication devices (modems, cables, ports), communication protocols and communication software enabling the transmission of data through networks.

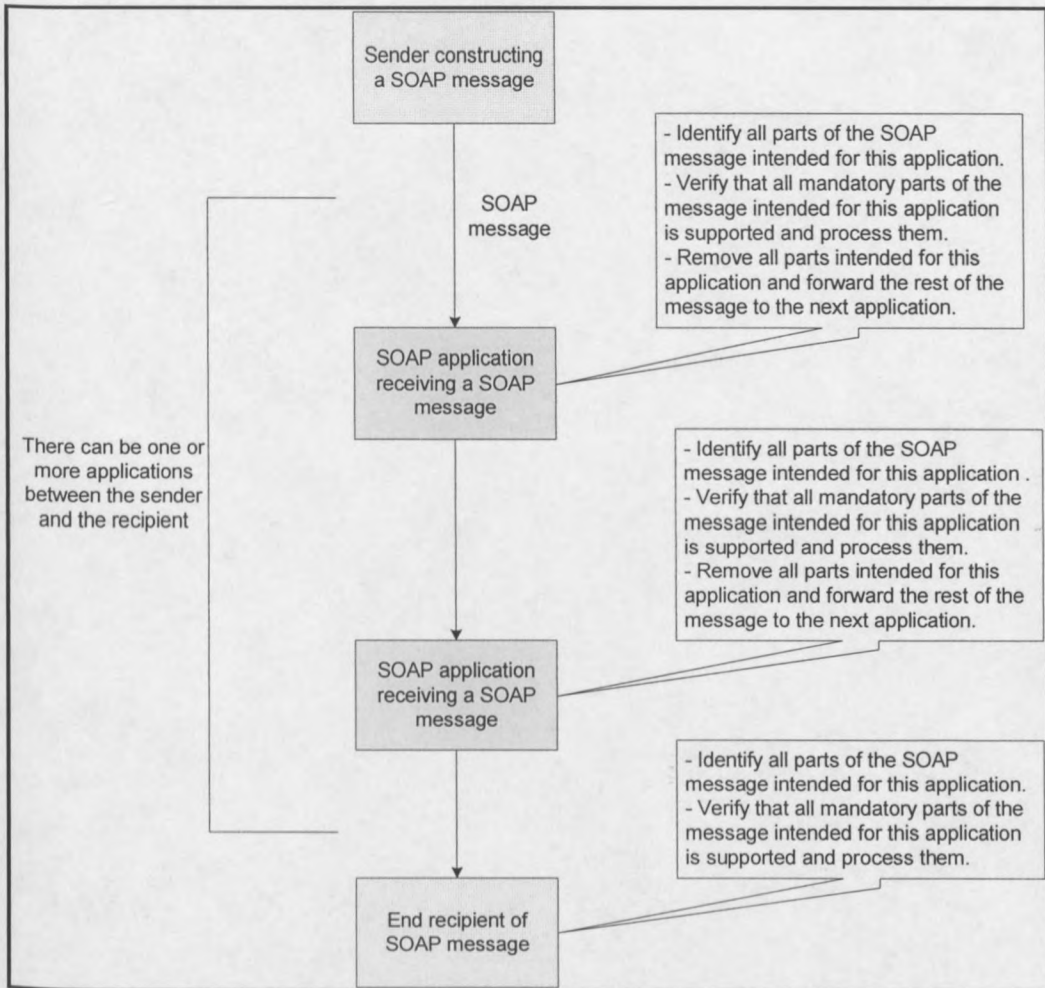
### **5.3.1 Simple Object Access Protocol (SOAP)**

“The Simple Object Access Protocol (SOAP) is a lightweight protocol for exchanging structured and typed information between similar parties in a decentralized, distributed environment using XML.” It allows objects of any kind, on any platform and in any language to cross-communicate.

SOAP enables the building of applications by remotely invoking methods on objects. It also removes the requirement that two systems must run on the same platform or be written in the same programming language. Instead of invoking methods through a proprietary binary protocol, a SOAP package uses XML for making method calls. All information between the requesting application and the receiving object is sent as tagged data in an XML stream over HTTP. From a web service’s point of view, SOAP may be implemented as either a client or a server.

The SOAP architecture consists of three parts, an envelope defining a framework for describing what is in a message and how to process it, a set of encoding rules defining a serialisation mechanism that can be used to exchange instances of application-defined data types and a convention for representing remote procedure calls and responses. A SOAP message is thus a XML document conforming to these rules and containing a mandatory SOAP envelope, an optional SOAP header and a mandatory SOAP body. The ‘envelope’ is the top element of the XML document representing the message, the ‘header’ a generic mechanism for adding features to a SOAP message in a decentralized manner without prior agreement between the communicating parties and the ‘body’, which is a container for mandatory information intended for the ultimate recipient of the message. [[www.w3.org/TR/SOAP](http://www.w3.org/TR/SOAP)]





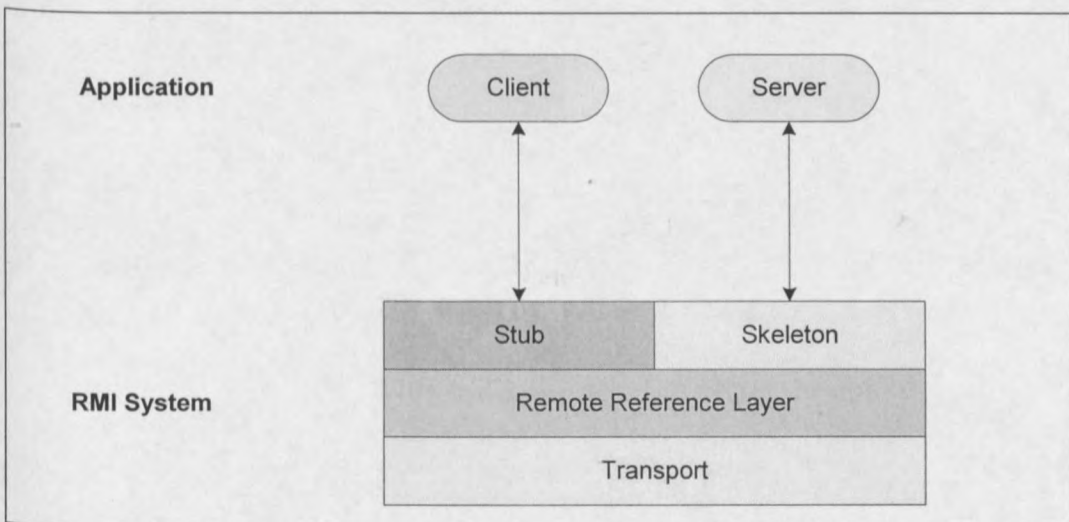
**Figure 6 SOAP Architecture overview**

### 5.3.2 Remote Method Invocation (RMI)

Remote Method Invocation is one of the processes by which a client may execute remote method calls on a server using a transparent interface. A client is supplied with the interface describing methods which are available from the remote server, but all implementations are left to the server side. In reality the details of implementing the remote methods are abstracted from the client software. The connection between the client and server of a RMI system is a layered connection operated by the RMI subsystem of the two virtual machines involved and is transparent to most developers of simple applications. Method invocation and data passing is handled via skeleton

and stub classes. A stub for a remote object is the client-side proxy for the remote object and implements all the interfaces that are supported by the remote object implementation. The skeleton for a remote object is a server-side entity that contains a method, which dispatches calls to the actual remote object implementation. The skeleton and stub classes are generated automatically through a RMI compiler and through communication using object serialisation and Transfer Control Protocol /Internet Protocol (TCP/IP). When a client tries to execute a method provided by the server, the client uses the stub class of the method provided by the server, which executes the actual implementing class of that service, thus remotely invoking a method. Except for the skeleton and stub classes, RMI is further made up of two layers, the Remote Reference Layer (RRL) and the Transport Layer (TL). The RRL transmits data to the transport layer via the abstraction of a stream-oriented connection, while the TL takes care of the implementation details of the connections.

“RMI is one of the ways in which Java supports distributed computing. It provides a way for Java programmes to interact with classes and objects working on different virtual machines on other network computers. From a client perspective, objects on the server may be referenced as if they were local.” [[www.java.sun.com](http://www.java.sun.com), [www.acm.org](http://www.acm.org)]



**Figure 7 RMI Architecture**

### 5.3.3 Remote Procedure Calls (RPC)

Remote Procedure Calls (RPC) is a powerful technique for constructing distributed client-server based applications. Other than RMI, which enables one to interoperate directly with a Java object, RPC enables the programmer to use standalone methods across a network. The advantage of RPC is in its transport independence. "RPC isolates the application from the physical and logical elements of the data communication mechanism and allows the application to use a variety of transports." As a result of this RPC allows almost any kind of application intercommunication, as the transport protocol can be HTTP.

When a client programme is compiled, the compiler creates a local stub (see section 6.2.2) for the client portion and another stub for the server portion of the application. When a client requires a remote function, these stubs are invoked and a request is sent to the server while the client waits. The thread handling this request is blocked from further processing until either a reply is received, or it times out. When the server receives the request, it invokes its stubs that perform the requested service and sends the reply to the client (see Figure 8). After the RPC call is completed, the client programme continues. [www.cs.cf.ac.uk, Java and XML, Brett McLaughlin,2000]

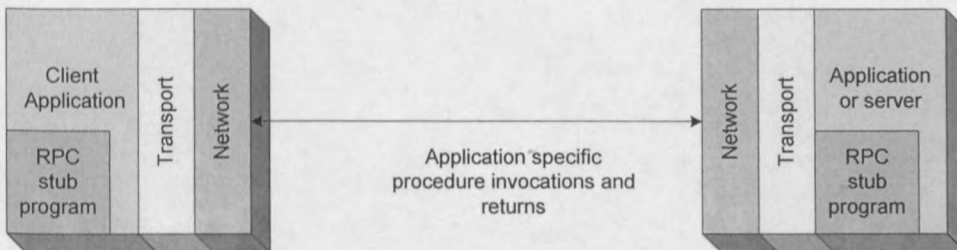


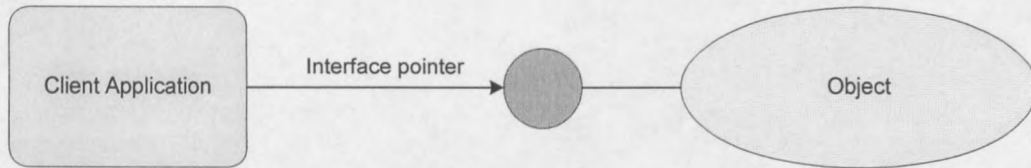
Figure 8 Remote Procedure Calls

### **5.3.4 Component Object Model (COM)**

The Component Object Model (COM) is a software architecture designed by Microsoft that allows components made by different software vendors to be combined into a variety of applications. A component is a reusable programme building block that can be combined with other components in the same or on other computers, in a distributed network, to form an application. A component can thus be anything from a single button in a graphical user interface to an interface for a database manager. COM provides the underlying layer that forms the foundation for higher level software services and was developed to:

- Define a binary standard for component interoperability.
- To be programming language-independent.
- To be available on multiple platforms (Microsoft Windows, Microsoft Windows NT, Apple Macintosh, UNIX).
- To provide for extensive evolution of component-based applications and systems.
- To be extensible.

The only point of contact between a client and a COM object are pointers to a set of interfaces defined for the specific COM object (Fig. 5). COM defines a binary structure for these interfaces. The structure provides the basis for interoperability between software components written in arbitrary languages. As long as a compiler can reduce language structures down to this binary representation, the implementation language for clients and COM objects does not matter, as the point of contact is the run-time binary representation. COM objects and clients can thus be coded in any language that support Microsoft's COM binary structure.



**Figure 9 Client using COM object through an interface pointer**

The main features deployed by COM are:

- Communications between components, even across process and network boundaries.
- Shared memory management between components.
- Error and status reporting.
- Dynamic loading of components.

COM is a general architecture and can be used by any software developer developing applications for data transfer or data storage. [[www.Microsoft.com](http://www.Microsoft.com)]

### **5.3.5 Distributed Component Object Model (DCOM)**

The Distributed Component Object Model (DCOM) is a low-level extension of COM and the main object technology used within Microsoft ActiveX, ActiveX being a higher level application service allowing components to be embedded in websites. “While COM processes could run on the same machine but in different address spaces, the DCOM extension allows processes to be spread across a network.” As a result of DCOM being an ActiveX technology, it also works natively with Internet technologies like TCP/IP, Java and HTTP, enabling business applications to work across the Web. DCOM further enables distributed Java, without requiring any communication-specific code or add-ons. [[www.Microsoft.com](http://www.Microsoft.com)]

### **5.3.6 Java™ 2 Platform, Enterprise Edition (J2EE)**

The Java technologies consist of three editions; the Java 2 Platform Standard Edition (J2SE), the Java 2 Platform Enterprise Edition (J2EE) and the Java 2 Platform Micro Edition (J2ME). The J2EE edition is aimed at enterprise applications, making them easier and simpler to create. This is achieved by:

- providing standardised, modular and re-usable components Enterprise JavaBeans™ (EJB™) on which these applications can be based.
- providing a complete set of services to these components.
- handling many details of application behaviour automatically.

The J2EE makes use of many of the Java 2 Platform Standard Edition features such as:

- Portability.
- JDBC API for database access.
- CORBA technology for interaction with existing enterprise resources.
- Security models.

In addition to these features J2EE also provide full support for Java Servlets API, Java Server Pages™ and XML technology, making network communication and e-business applications simpler. [[www.java.sun.com](http://www.java.sun.com)]

### **5.3.7 Microsoft .NET**

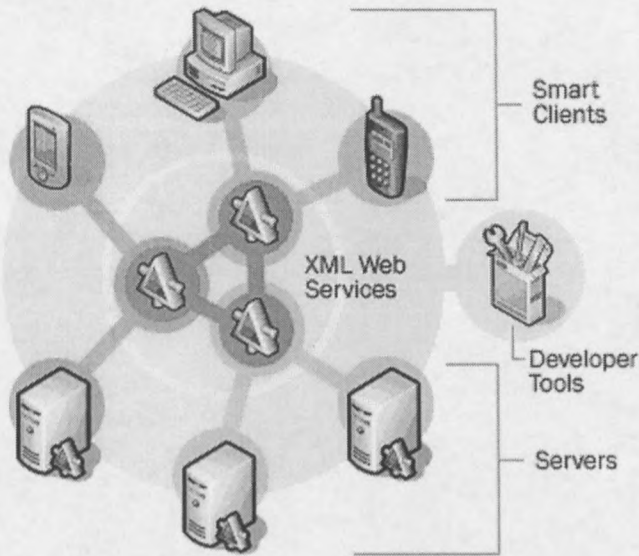
The .NET initiative from Microsoft is a platform for building and running the next generation of distributed applications via the Internet. This means that the Internet is used as the basis for a new operating system, reducing the client's need for hardware by making data available from the Internet.

.NET provides programmers with

- a programming model that enables the development of XML web services and applications.
- a set of XML web services such as Microsoft .NET My Services.
- a set of servers such as SQL Server™ and BizTalk™ Server.
- client software such as Windows XP and Windows CE.
- tools, such as Visual Studio® .NET, to develop XML Web services.

XML web services are small, building block applications written in XML. It enables communication between different XML web services and other larger applications across the Internet. These web services can be used in different ways (Figure 10):

- Client to client- 'smart' clients or devices can host and apply XML web services allowing data to be shared.
- Client to server- data from a server can be shared to a desktop or mobile device via the Internet by using XML web services.
- Server to server XML web services provide a common interface between existing applications in an independent server environment.
- Service to service- XML web services can work in sequence to provide a more complex data operation.
-



**Figure 10 .NET and XML web services**

[<http://www.microsoft.com/net> , [www.gotdotnet.com](http://www.gotdotnet.com)]

### 5.3.8 Hypertext Transfer Protocol (HTTP)

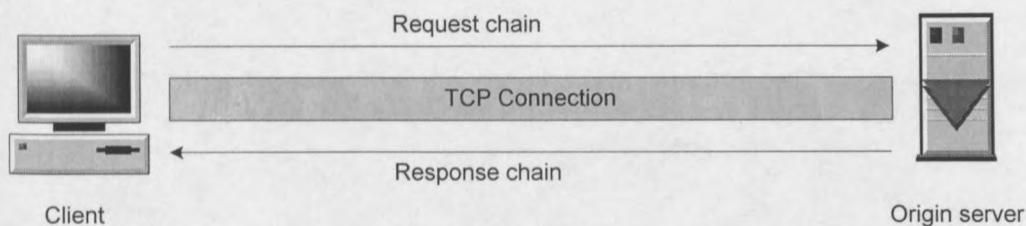
“The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative and hypermedia information systems. It is a generic stateless object-oriented protocol.” Stateless, meaning that for a typical implementation a Transmission Control Protocol (TCP) connection will be created between the client and the server for each transaction and will terminate as soon as the transaction is completed. HTTP may be used for many similar tasks, such as name servers and distributed object-oriented systems. The data transferred by HTTP can be any Internet accessible information such as images, audio, plain text or hypertext.

HTTP makes use of TCP to provide reliability, but still remains stateless as each transaction is treated independently. It further allows an open-ended set of methods that indicate the purpose of a request. It then builds on the discipline of references for indicating the resource to which a method is to be applied. These references include:



- the Uniform Resource Identifier (URI)
- the Uniform Resource Location (URL)
- the Uniform Resource Name (URN)

When a client wants to retrieve information from the Internet, the client creates a TCP connection to the server on which the information resides. This server is called an origin server. The client then issues an HTTP request containing a specific command or method, a URL and a Multipurpose Internet Mail Extension (MIME) type message containing request parameters and information about the client. When the server receives the request, it performs the request if possible and sends a HTTP response back to the client. The response consists of status information, information about the response itself and possible body content. After this, the TCP connection is closed and the transaction is complete (Figure 11).



**Figure 11 HTTP Operation**

Another important feature of HTTP is that it is flexible in the formats that it handles. For example, when a client issues a request to a server, it may include a list of prioritised formats that the client browser is able to handle. [[www.w3.org](http://www.w3.org) , Data and Computer Communications, William Stallings, Prentice Hall, 2000]

### 5.3.9 Summary of Communication Technologies

The communication technologies discussed above is summarised in Table . It provides a reference for choosing which technology to use under which circumstances.

**Table 5 Communication Technologies**

Technology	Controlling body	Data Transport format	Platform Independent	Programming Language	Connection protocol	Implementation side	Other technologies used	Implemented as	Used for
<b>SOAP</b>	World Wide Web Consortium	XML conforming to SOAP rules	Yes	Any	HTTP	Client and server side	RMI	Client or server one way messaging exchange service.	Exchanging structured and typed information between peers.
<b>RMI</b>	Sun Microsystems	Marshallled Objects, predefined data types	Yes	Java	TCP/IP	Server side, objects to be used resides on the server.	TCP/IP -	Client/server model	Distributed computing.
<b>RPC</b>	Unix	Operating system binary data streams	Yes	Any	Any	Server side, methods to be invoked resides on the server.	Any connection technology-	Client/server model	Distributed computing, ignoring the details of the network interfaces.
<b>COM</b>	Microsoft	Local application to application binary streams	No	Any language supporting the COM binary structure.	TCP/IP	All COM objects run inside a server.	Makes use of Remote Procedure Calls	Specification and implementation for a framework for integrating components	Providing a framework for integrating components
<b>DCOM</b>	Microsoft	Network application to application binary streams	No	Any language supporting the COM binary structure.	TCP/IP and HTTP	All DCOM objects run inside a server.	Distributing computing environment	Communication technology for COM objects	Allowing network based component interaction.
<b>J2EE</b>	Sun Microsystems and a collaboration of software vendors	SOAP	Yes	Java	TCP/IP and HTTP	Client, server or services.	JDBC, RMI, Java API for XML-based RPC (JAX-RPC), J2EE Connection Architecture (JCA)	Standard for developing multitier enterprise applications.	Developing distributed enterprise applications.
<b>Microsoft .NET</b>	Microsoft	SOAP	No	Any	TCP/IP and HTTP	Client, server or services.	RMI, DCOM, Microsoft Message Queuing (MSMQ)	Client, server or services.	Building and running distributed applications.
<b>HTTP</b>	World Wide Web Consortium	Plain text, hypertext, audio, images, any Internet-accessible information.	Yes	Any	HTTP	Transfer between client and server.-	TCP/IP	Transfer protocol	Any client/server application involving hypertext

## **5.4 Internet technologies**

The Internet is a global network of networks, or a networking infrastructure, allowing any computer to communicate with any other computer as long as both are connected to the Internet. Internet technologies are the technologies making this possible. Although Internet technologies consist of different networks, computers, software, cables and modems, only the technologies to be used in this thesis will be reviewed, thus servers and servlets. These technologies however can also be used in other fields and are not restricted to Internet applications.

### **5.4.1 Servers**

A server is an application programme that accepts connections from other programmes in order to service requests and send back responses. Many different servers are available and the choice to use a specific server will depend on the functionality required and cost considerations.

#### **5.4.1.2 Apache Tomcat server**

The Apache Tomcat server is the reference implementation for the Java Server Pages (JSP), a technology allowing tags and inline Java code in a normal HTML document, and the Java Servlet specifications and therefore is the most standards compliant Java server available.

Tomcat is a Servlet container (see 5.4.2 Servlets), which means that Java Servlets can be used within it to enable programmers to use the Servlet. An example of this is the dynamic creation of web pages.

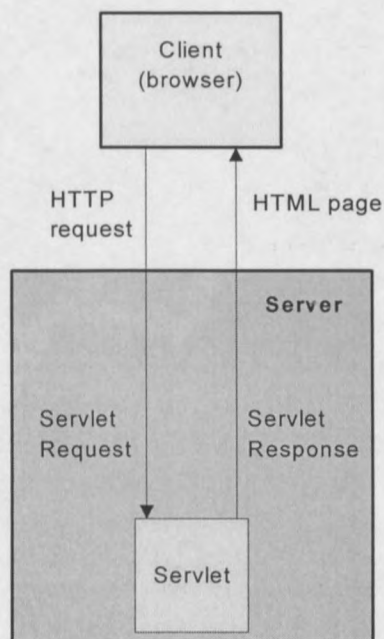
For documentation on installing, configuring, creating web applications and managing the Tomcat server see Appendix C.

## 5.4.2 Servlets

Servlets are server side programmes written in Java and are responsible for handling requests received from the client. Once a server receives a request with a specific servlet name connected to that request, the server will hand the request directly to the servlet. The servlet will then use local information, parameters passed from the requesting client and other resources to construct a Hypertext Markup Language (HTML) page or any other desired format. This result will then be sent to the server, which will send it back to the client.

Except for returning useful information for web pages, servlets can also be used to listen for remote connections from applets and return real time data to the client application. This technology allows programmers to develop platform independent extensions that can be integrated simply with an existing web server.

The big advantage in using servlets is that it is much faster, easier to use and more secure than traditional standards for interfacing external applications. [Advance Techniques for Java Developers, Daniel J. Berg, J. Steven Fritzinger, Wiley Computer Publishing, 1999]



**Figure 12 Client - Server - servlet relation**

### 5.2.5 Java Server Pages (JSP)

The Java Server Pages (JSP) provides a method for the dynamic creation of web page content and separate the dynamic content from the static content of a web page. A JSP page is created using HTML and its associated tools, but when dynamic content must be inserted it is enclosed in special tags as shown below.

```
<BODY>
<H1>Using JSP</H1>
<USER>Welcome back,
    <% out.println(getUserNameFromCookie(request)); %>
    To access your account settings, click
    <A HREF="Account-Settings.html">here.</A>
</USER>
</BODY>
```

When the JSP file is completed it is saved as a .jsp file and stored where usual web pages would be kept. When the first client request the JSP page, it is converted to a normal servlet, compiled and loaded. The static HTML is then printed to the output stream of the servlet and displayed to the client.

## **5.5 Database technologies**

Databases are used to store and organise data in such a manner that it is easily traceable and accessible for application programmes requesting the data. All databases have the same in common; the storing, managing and manipulation of data. Depending on the database manufacturer, a specified query language is used to query the database in a standard way.

### **5.5.1 Database types**

Many different types of databases exist and the criteria for choosing a database depends on the amount of data that must be handled, the structure of the data as well as how frequently the data will be accessed.

Types of databases include flat-file databases, relational databases and object databases.

#### **5.5.1.1 Flat-file Databases**

Flat-files are files containing data with no structured relationships and requiring additional information to be interpreted. A Flat-file Database is a database system containing a collection of flat-files, each stored in a single table.

Flat-file Databases are used when small amounts of data, that do not require frequent updating and must be readable by humans, are handled. These databases consist mainly of sets of strings in one or more files that can be parsed (see 5.6.5 Parsers) to obtain the data they contain.

The storage of more complex data in a flat-file database is possible, but becomes complicated and is more time consuming and requires more processing power than when a relational database is used for complex data storage. When complex data in a flat-file database is parsed (see 5.6.5 Parsers), it is also likely to become unreadable and un-editable. The main problem with flat-file databases is that they are easily

corrupted, as there are no inherent locking mechanisms that detect when a file is in use or is being modified. This means that locking and unlocking of files must be handled in the script and that it is possible for two or more processes, competing for a file, to delete the file. The timing of locking and unlocking of files thus become very important the busier the database becomes.

Database Management Systems (DBMS) are systems using binary flat-files for the storage of data. They allow for a set of strings to be associated with a key value, thus allowing for faster retrieval of data. The DBMS essentially adds more functionality and better sorting of the binary flat-files it uses in comparison to a script controlled flat-file database. The advantage of using DBMS is that a programmer does not have to worry about how the data is stored or how the values must be retrieved. This is all handled by the DBMS. The creation, editing or deletion of data is also made simple by normal methods being provided for once a connection to the database is established. Another advantage is that data retrieval is faster as data is stored in such a manner that allows faster location of key-value pairs.

The problem of file locking and unlocking however, is still a problem when using DBMS. [[www.websiteowner.info](http://www.websiteowner.info)]

### **5.5.1.2 Relational Databases**

Relational databases are databases with relations between the different tables it contains. The data in such databases is ordered in a much more logical way than in flat-file databases and is often used to represent a set of objects, with each column in the table representing an attribute of this object. A table “Cars” for instance could have the columns: *type*, *model*, *engine*, *colour* and *cost*, with each different row constituting a different car. The “relation” for the database is due to the “relation” the different tables can have. An *engine* can be cross-referenced to a table “Engines”, containing all the specifications for different engines.

Normalisation, a process of dividing the data in a database into two or more tables and specifying a relationship between the tables, is used to minimise redundancy of

relational databases. The idea is to isolate data so that changes to data can be made only to one table and be propagated to all the tables relating to the changed table.

A big advantage of relational databases over flat-file databases is that when a relational database is well designed, there should be no duplicate data. This has considerable advantages with regard to file size and maintaining data integrity.

Another advantage is the built in functions provided by the databases for retrieving, sorting and editing of data. Because these functions are able to sort and filter the data they only send back result sets of the required data. This means that sorting and filtering of data does not have to be handled on script level, saving the programmer time and effort. [www.websiteowner.info]

### **5.5.1.3 Object orientated database**

Object orientated databases (OODB) are databases that support the modelling and creation of data as objects. It is designed to work well with object orientated programming languages such as Java and C++ and should be used when Java, C++ or other object orientated programming language objects must be stored in a database. An advantage of OODB is its ability to give high performance when working with complex data.

As to date, there is no specification as to what constitutes an object orientated database system. The Object Data Management Group (ODMG) defines the criteria for an OODB in their paper '*The Object-Oriented Database Manifesto*, by Malcolm Atkinson and others' as follows:

*"An object-oriented database system must satisfy two criteria: it should be a DBMS, and it should be an object-oriented system, i.e., to the extent possible, it should be consistent with the current crop of object-oriented programming languages. The first criterion translates into five features: persistence, secondary storage management, concurrency, recovery and an ad hoc query facility. The second one translates into eight features: complex objects, object identity,*



*encapsulation, types or classes, inheritance, overriding combined with late binding, extensibility and computational completeness.”*

[An Exploration of Object Oriented Database Management Systems, Dare Obasanjo, 2001, [www.odbmsfacts.com](http://www.odbmsfacts.com), [www.odmg.org.standard/](http://www.odmg.org.standard/) ]

#### 5.5.1.4 Summary of database types

Following is a summary of the different databases commonly used. The table provides information for assisting in the selection of a database for specific purposes, by providing information on usage, advantages and disadvantages.

**Table 6 Database type comparison**

Database type	Data stored as	Used for	Advantages	Disadvantages
<b>Flat-file</b>	Sets of strings in one or more files.	Storing of small amounts of data not needing frequent updating and that must be human readable.	Human readable. Small. Well suited for storing simple lists and data values.	No locking mechanisms cause files to be easily corrupted. Difficult to store complex data. Data duplication.
<b>Relational</b>	Tables	Storing and manipulation of relational data.	Functions provided for retrieving, sorting and editing of data. No duplicate data. Deals efficiently with large amounts of data.	Require detailed documentation of created databases, knowledgeable personnel and higher hardware and software requirements.
<b>Object</b>	Objects	Storing of object orientated programming language objects.	High performance on complex data. Fast development. A query language is not needed. Whole system can easily be illustrated in a UML diagram.	Objects typically tied to a specific language, thus only accessible from that language. Queries on the data are dependent on the design of the system. Any changes to the database will need the system to be recompiled.

## 5.5.2 Database processing tools

Database processing tools provide mechanisms for applications to access and query databases on a remote or local system. Many different tools exist for this purpose and it is open to the developer to decide which one will suit their application the best.

### 5.5.2.1 Java Database Connectivity (JDBC)

The Java Database Connectivity (JDBC) is a platform independent Application Programmers Interface (API), allowing programmers to programme code without having to worry about which database it is being connected to. It is however still possible to make database specific calls using JDBC. For the JDBC API to be platform independent it makes use of a drive manager that is responsible for dynamically maintaining all the driver objects the database queries will need. The loading of these driver objects and registration to the drive manager can be forced by using methods provided for this purpose by JDBC. To connect to and query a database using JDBC, the following steps must be executed:

1. Load the required drivers as specified by the database vendor with the method `Class.forName("Driver to be used");`.
2. Connect to the database after the driver has been loaded by including the following line of code:

```
Connection con = DriverManager.getConnection(url, "username", "password");
```

The URL to be provided depends on the driver being used, for example if you use the JDBC-ODBC Bridge driver the URL would start with `jdbc:odbc:`, and would usually be followed by the data source name. If the data source is called "Information", the URL would be `jdbc:odbc:Information`. The username and password will be the name and password used to log into the database management system.

3. Once the drive manager recognises a given URL it returns an open connection to the database that can be used to create JDBC statements. These statements can then be used to pass SQL statements to the database management system for querying the database.

[[www.java.sun.com](http://www.java.sun.com) , Thinking in Java, Bruce Eckel, Prentice Hall, 1998]

### **5.5.2.2 Open Database Connectivity (ODBC)**

Open Database Connectivity (ODBC) is a database access standard that permits applications to connect to a variety of external database servers and other sources of data. The first ODBC driver was built by Simba technologies in 1994 under contract from Microsoft and marked the beginning of standards-based data access on every desktop. Before, ODBC data access was specific to each database vendor. ODBC changed this by providing an open interface for cross-platform data connectivity, giving business intelligent tool vendors a standard so their report writers and query builders could have a single interface to multiple data sources. [[www.simba.com/](http://www.simba.com/)]

### 5.5.2.3 Summary of database processing tools

The controlling bodies and uses of the different database processing tools are summarised in Table 7.

**Table 7 Database processing tools**

Technology	Controlling Body	Used for
JDBC	Sun Microsystems	Programming code without knowing the database being connected to, made possible by JDBC Platform independency.
ODBC	Simba Technologies under contract from Microsoft	Connecting applications to external database servers and data sources.

## **5.6 Document manipulation techniques and tools**

A document, in Information Technology terms, is a file created by a word processor or other application. Each document has a specific style, syntax and structure and can contain objects like graphics or charts. The content, style and structure of a document is accessible and can be updated dynamically by programmes or scripts. When working with markup languages, documents contain pure textual data, consisting of text marked up with tags (<a tag>).

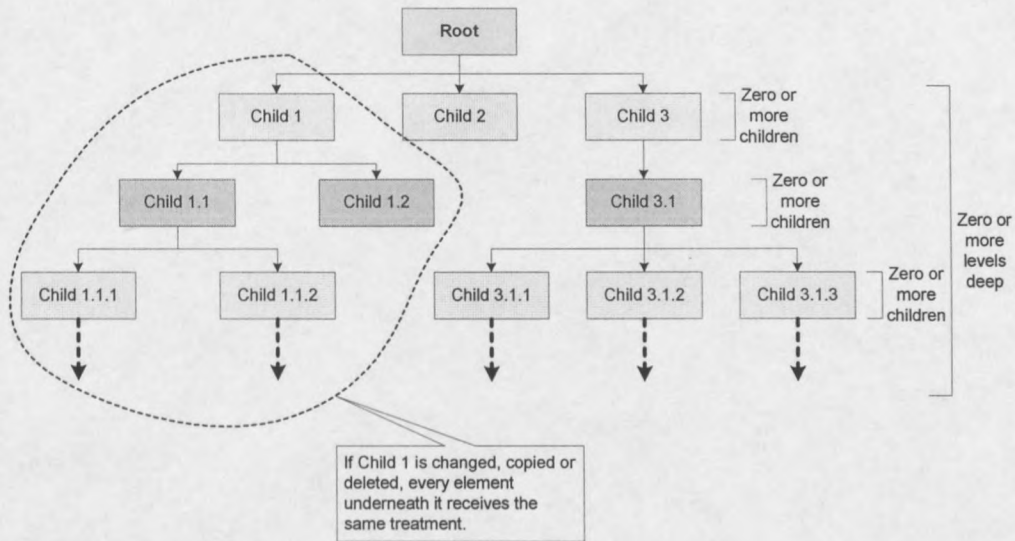
Document manipulation tools are programmes, scripts or technologies used to create new documents or to transform, modify or update existing documents.

### **5.6.1 Trees**

A tree structure is a hierarchical representation of data used to organise, access and modify data in an easy manner (Figure 13). It consists of a root element at the top of the structure and zero or more children. Each of the children can consist of zero or more children and this can continue to an arbitrary level. All children are elements and all elements can contain elements, textual data and attributes.

Iterations, a process of repeating the same procedures, for instance to step through a data structure, are used to step through trees and at any point in this process an element or node can be modified, styled, ignored, copied or have something done to it.

An advantage of using trees is that it allows the grouping of documents to be maintained. If, for instance Child 1 in Figure 13 is copied, deleted or moved, all of its children and their children and their children until the end of the structure receive the same treatment as Child 1. [Java and XML, Brett McLaughlin, O'Reilly&Associates, Inc., 2000]



**Figure 13 Document Tree structure**

### 5.6.2 Document Object Model (DOM)

The Document Object Model (DOM) is a cross-language, cross platform specification for representing the content and structure of documents. It is a World Wide Web consortium recommendation (W3C) and is organised in levels. Level one is concerned with document content, detailing the functionality and navigation of content and support XML 1.0 and HTML 4.0. Level two builds on level one by adding functionality for specific content models. These include XML with namespaces, stylesheets describing how to format each element in a document called Cascading Style Sheets (CSS) as well as more sophisticated tree manipulation events. The third and last level is still under development. This level builds on the previous two levels by finishing the support for XML 1.0 namespaces and will add functionality for validation, for loading and saving of documents and for exploring mixed markup vocabularies (embedded DOM). It will also support XPath.

[Java and XML, Brett McLaughlin, O'Reilly&Associates, Inc., 2000, [www.w3c.org](http://www.w3c.org) ]

### **5.6.3 Java Document Object Model (JDOM)**

The Java Document Object Model (JDOM) is a Java-centric, high-performance Application Programmers Interface (API). It is used for creating a Java representation of a XML document in such a manner that writing, manipulation and reading of the data is fast and efficient. JDOM is an alternative to both DOM and the Simple API for XML (SAX) although it integrates well with both of these.

JDOM is namespace aware, allowing each element to be declared in a specific namespace, supports validation through Data Type Definitions (DTD) or XML Schema Definitions (XSD), makes use of the Java 2 collection classes and uses XML parsers to build documents.

JDOM is also easy to learn and understand, as it was developed with the Java programmer in mind and follows proven Java design patterns. In general, XML documents can be seen as a whole, with any member of the document available at any time. JDOM constructs are created through direct object instantiation and methods for the easy editing of XML documents are provided and the normal Java Input Output (IO) classes are used for inputting and outputting of data.

[Java and XML, Brett McLaughlin, O'Reilly & Associates, Inc., 2000, [www.jdom.org](http://www.jdom.org)]

### **5.6.4 Simple API for XML (SAX)**

The Simple API for XML (SAX) defines a framework for parsing XML data. It defines events, like document beginnings and ends, during parsing (see 5.6.5 Parsers), allowing full control over these parts of the parsing process. SAX also defines a set of error and warning messages, used for handling invalid situations that can occur during parsing. It is important to notice that SAX only provides a framework for parsing, and is not a parser itself. It provides methods to specify the parser to use and events to monitor the parsing process. [Java and XML, Brett McLaughlin, O'Reilly & Associates, Inc., 2000, [www.saxproject.org](http://www.saxproject.org)]

### **5.6.5 Parsers**

Parsers are programmes responsible for dividing an XML document being read into an application into individual elements, attributes and other tokens. While parsing, the parser will pass the contents of the document token by token to the application. If at any point the parser encounters a violation of the XML rules, it will stop parsing the document and return an error message to the application. This kind of parser only checks whether the document is well formed, verifying that all the XML rules have been adhered to. XML documents can optionally refer to a Document Type Definition (DTD) or XML Schema Definition (XSD), describing the structure and logic of the document. The DTD or XSD can either be included into a XML document or be referenced from within the document.

Parsers verifying that XML document adhere to a DTD or XSD is called validating parsers and violation of these rules are called validity errors. Once a validity error occurs the parser sends an error message to the application programme, which can then decide if it wants to continue parsing.

[XML and Java, Hiroshi Maruyama, Kent Tamura, Naohiko Uramoto, Addison Wesley Longman, Inc., 1999, Java and XML, Brett McLaughlin, O'Reilly & Associates, Inc., 2000]



## CHAPTER 6

### Project Information Management

This chapter deals with the different documents and forms used to convey data between the different parties of a project. It describes the purpose and data of each document and provides the basis for the next chapter, Project Information Procedures. The importance of this chapter is in that it provides the necessary information about what data is important for project management and when in the lifecycle of the project it is used.

#### 6.1 Project management documents

Many different documents or forms for the management of project data exist, each containing data for a specific aspect of a project. As a project proceeds in its life, so different forms become necessary to capture the newly acquired data, often overlapping with previous forms' data. What follows is a selection of some of the most used and more important forms for project management.

##### 6.1.1 Application Form

In general application forms are used when an organisation or entity wants to apply for something. This could be anything from funds for a project to an application for construction of a house. A variety of different formats of application forms exist, depending on whom you're dealing with, but they all contain the same basic information. This includes fields like; the applicants information, project details and motivation, a factual summary of the project, funds applied for, beneficiaries and so forth. This is the first form that will be submitted to an authority in a project's life cycle. [[www.cmip.co.za](http://www.cmip.co.za)]

### 6.1.2 Business Plan

A business plan, also known as a feasibility study, is a written description of a project or business' future. It therefore is a document describing the plan of what needs to be done and how to achieve it.

In general, a business plan conveys the goals of a project, the strategies needed to meet them, potential problems that may confront the project and ways to solve them. It will also include the organisational structure of the business (including titles and responsibilities) and the amount of capital required to finance the project. A business plan usually follows generally accepted guidelines and can be divided into three main parts:

- **A business concept:** describing the industry, the project structure and the plans to make the project or business a success.
- **A target group section:** describing who will benefit from the project and why the project is important and other related questions.
- **A financial section:** containing the income and cash flow statement, balance sheet and other financial ratios.

[[www.cmip.co.za](http://www.cmip.co.za)]

### 6.1.3 Tender Document

Tender documents are documents describing all the specifics of a project. In general, depending on the contract, these will include:

- Conditions of tender.
- General conditions of contract.
- Special conditions of contract.
- Project specifications.
- Particular specifications.
- Schedule of quantities.

- Tender, appendix and annexe.
- Agreement and deeds of surety.
- Surveys.

The tender document is handed out to any interested party wishing to compete for the tender. The tender document must then be completed and handed back to the engineer for examination. After examination the engineer can make suggestions to the client as to which tender to accept.

[GFJ Incorporated, Consulting Engineers and Project Managers]

#### **6.1.4 Bill of Quantities**

The Bill of Quantities is a document drawn up for tender purposes. The document contains all the items necessary for a project, divided into sections as specified by SABS 1200. In general these include:

- Preliminary and general items.
- Site clearance.
- Sections specific to the project for example sewers, water mains, road works, concrete works and so forth.

A typical example of a Bill of Quantities is given below.

Item	Payment	Description	Unit	Quantity	Rate	Amount
	Clause					
		SCHEDULE A: PRELIMINARY AND GENERAL				
	SABS 1200 A	PRELIMINARY AND GENERAL				
A.1	8.3	FIXED-CHARGE AND VALUE- RELATED ITEMS				
A.1.1	8.3.1	Contractual requirements	Sum	1		
	8.3.2	Establish facilities on the Site:				
	8.3.2.1	a) Facilities for Engineer (SABS 1200 AB)				
A.1.2		One office complete with carports as described in PS AB 3.2	Sum	1	4000	4000
A.1.3		2 Name boards	Sum	1	2500	2500
	8.3.2.2	b) Facilities for Contractor				
A.1.4		Offices and storage sheds	Sum	1	4250	4250
A.1.5		Workshops	Sum	1	1000	1000
A.1.6		Laboratories	Sum	1	12000	12000
A.1.7		Living accommodation	Sum	1	500	500
A.1.8		Ablution and latrine facilities	Sum	1	1200	1200
A.1.9		Tools and equipment	Sum	1	5600	5600
A.1.10		Water supplies, electric power and communications	Sum	1	8500	8500
A		PRELIMINARY AND GENERAL				
		SCHEDULE B: SITE CLEARANCE				
B.1	SABS 1200 C	SITE CLEARANCE				
	8.2.1	Clear and grub	ha			
B.1.1		Site for earth works	ha			
B.1.2		Fill area on erven	ha			
B.1.3		Street reserves	ha	1.3	5000	6500
B.1.4		Pipelines	m	2155	50	107750
B.1.5	8.2.6	Clear hedge/fence or both	m			
	PS C 8.2.7	Dismantle and remove pipelines, electricity transmission lines, cables, etc				
B.1.6		Pipelines, dia up to 300 mm	m	400	66	26400
B.1.7		Pipelines, dia between 300 mm and 600 mm	m	200	120	24000
B.1.8		Pipelines, dia between 600 mm and 900 mm	m	150	225	33750
B.1.9		Cables	m	600	10	6000
B.1.10	PS C 8.2.11	Remove existing walls, kerbs, etc	m	1400	12	16800

As can be seen above, the Bill of Quantities contains seven columns: *Item*, *Payment Clause*, *Description*, *Unit*, *Quantity*, *Rate* and *Amount*. The *Item* column is for reference to a specific item, the *Payment Clause* column refers to the payment clause in SABS 1200 or a special item clause in the tender document. The *Description*

column is used for a short item description and the remaining four columns are for quantity and amount purposes.

Also contained in the Bill of Quantities is a summary of all the sections included in the document, consisting of the subsection names and the total amounts for the subsections. [GFJ Incorporated, Consulting Engineers and Project Managers]

### **6.1.5 Contractors Appointment Letter**

The letter of appointment confirms the employment of the contractor and states the conditions under which the contractor is employed and will usually, depending on the contract, include the following:

- A guarantee of 10% of the contract value redeemable when the contractor fails to complete the works.
- Insurance for the works as specified by the engineer.
- Proof of payment for the accident fund of the contractor's employees.
- Insurance against riots if specified by the engineer.
- That the work must commence within a certain time, usually 14 or 30 days.

[GFJ Incorporated, Consulting Engineers and Project Managers]

### **6.1.6 Payment Certificate**

A payment certificate in the engineering industry is a document issued by an engineer to a contractor to certify that work of a certain worth has been done. The document is submitted by the contractor to the client for payment. The payment certificate will typically contain:

- the words 'Certificate of Payment to the Contractor' stated prominently on the document.

- the name and address of the client and the contractor.
- the date of issue of the certificate.
- the amount paid to date.
- the amount payable.
- the balance due on the contract.
- the engineer and contractor's signature.

[GFJ Incorporated, Consulting Engineers and Project Managers]

### **6.1.7 Tax Invoice**

When goods or services are supplied, an invoice is issued to the client as proof of purchase. A tax invoice contains additional information that may not appear on an ordinary invoice and will usually include:

- the words 'tax invoice' stated prominently on the document.
- the date of issue of the tax invoice.
- the name or trading name of the supplier.
- the business number of the entity issuing the document.
- the VAT inclusive price of a taxable supply or service.
- a brief description of each item or service supplied.
- the name of the client.
- the address of the client.
- the quantity or volume of what has been supplied.

[GFJ Incorporated, Consulting Engineers and Project Managers]

### **6.1.8 Reports**

Reports are a means of conveying information between different parties. Many different types of reports exist, depending on the nature of the information being conveyed. Typical reports include:

- Status reports
- Progress reports
- Expenditure reports

All of these reports will include information on the following:

- Information specific to the general contract details e.g. Contract no., employer and contractor names, total expenditure to date, accepted tender amount, commencement date, report number and so forth.
- A summary of the data described by the report name, e.g. the progress report will contain data describing the progress of the project.

[GFJ Incorporated, Consulting Engineers and Project Managers]

### **6.1.9 Variation Orders**

Variation orders are used when aspects of a contract are altered, omitted or if something is added to the contract. Changes to the contract causing the issuing of a variation order is usually caused by unforeseen events, such as an estimate being totally wrong for practical reason during construction, or if the engineer thinks it fit to change some of the designs. Fields included in a variation order include:

- the words 'Variation order' stated prominently on the document.
- reference no., contract no. and date.
- the contractor's details.
- the employer's details.

- the nature of the variation including a description thereof and a layout of the associated costs.
- the total cost of the variation.
- the influence on the tender amount.
- the engineer and employer's signatures.

A variation order is issued by the engineer, confirmed by the employer and handed to the contractor to act accordingly. [GFJ Incorporated, Consulting Engineers and Project Managers]

### **6.1.10 Certificate of Practical Completion**

Certificates of practical completion are issued once contract work is such that it can be used for its designed purpose without danger and too much inconvenience to the user. Fields included in a certificate of practical completion are:

- the words 'Certificate of Practical Completion' stated prominently on the document.
- the name, address and reference number of the employer and contractor.
- the contract number, a summary of the contract and the data of practical completion.
- a certificate number.
- the clauses from the General Conditions of Contract specifying the conditions for practical completion as well as clauses for reducing of penalties for late completion.
- all the work completed.
- all uncompleted work.
- the engineer and contractor's signatures and the date of signing.

After a certificate of practical completion is issued, all the uncompleted work stated in the certificate must be completed before a certificate of completion can be issued.

[GFJ Incorporated, Consulting Engineers and Project Managers]



### **6.1.11 Form of surety release**

The engineer issues the form of surety release to the contractor once the retention time has lapsed and an inspection of the works revealed that the works is still satisfactory. Fields in this form include:

- the words 'Form of Surety Release' stated prominently on the document.
- the name and address of the person to whom the form is issued.
- the contract number and effective date.
- the contractual certificate reference number.
- the employer's name and reference number.
- the contractor's name, address and reference number.
- a contract description.
- the surety of guarantor address.
- the type of guarantee.
- the clauses from the General Conditions of Contract stating either that the guarantee can be released, reduced or not released.
- the engineer's signature and the date signed.

[GFJ Incorporated, Consulting Engineers and Project Managers]

### **6.1.11 Form of surety release**

The engineer issues the form of surety release to the contractor once the retention time has lapsed and an inspection of the works revealed that the works is still satisfactory.

Fields in this form include:

- the words 'Form of Surety Release' stated prominently on the document.
- the name and address of the person to whom the form is issued.
- the contract number and effective date.
- the contractual certificate reference number.
- the employer's name and reference number.
- the contractor's name, address and reference number.
- a contract description.
- the surety of guarantor address.
- the type of guarantee.
- the clauses from the General Conditions of Contract stating either that the guarantee can be released, reduced or not released.
- the engineer's signature and the date signed.

[GFJ Incorporated, Consulting Engineers and Project Managers]

### **6.1.12 Final certificate**

A final certificate states that the engineer, after the retention time has lapsed, is satisfied that the contractor has fulfilled all of his obligations as stated in the contract and that all works specified were completed and/or maintained. Provision is made for any unfulfilled obligations. This certificate is issued by the engineer to the contractor and marks the final payment of any outstanding payments as well as the release of guarantees or retention money. It also states that after this final payment the contractor will have no more claims on the employer except for any unfulfilled obligations that need to be fulfilled. Fields included in this certificate are:

- the words 'Final Certificate' stated prominently on the document.
- the certificate reference number.
- the name, address and reference number of the employer and contractor.
- a description of the contract, the contract number and the date the works commenced.
- the reference completion and final payment certificate numbers.
- the date the works was completed.
- the final contract price and the maintenance expiry date.
- the clauses from the General Conditions of Contract stating the conditions for the issuing of a final certificate.
- any unfulfilled obligations.
- the engineer and contractor's signatures and the date it was signed.

[GFJ Incorporated, Consulting Engineers and Project Managers]

**Note:** Images of the forms studied in this document cannot be reproduced here as per copyright requirements of the authoring organisation.

## CHAPTER 7

### Project Information Flow

This chapter deals with the ways in which project information is passed between different parties involved in a project. It includes the different procedures to be followed, from project application to project completion as well as the documents needed for each phase of a project. It provides an understanding of current implementations of Project Information Systems, which guided the design of a more advanced, compatible information system which was developed in this thesis.

#### 7.1 Project Procedures

Project procedures are all the different ways in which the different parts of a project's lifecycle are managed. It involves the interaction between an employer, an engineer and a contractor and generally consists of an initialising, a design and a construction phase, as indicated in Figure 14, taken from 'Civil engineering procedure, 5<sup>th</sup> edition, pg 72'. More complex models for modelling projects are available and the reader is referred to 'The Process protocol' model, defined and described in the ActionIT document 'A Generic Design and Construction Process Model for Projects in Government'. Typically a project will start off with an idea, initiated by a need. This idea will then be tested for feasibility and if viable, the design and construction phases will follow. The design phase includes the documentation, tendering and awarding of the contract. The information flow between the different phases is complex and consists of many documents and forms being passed between the different parties. Figure 14 shows some of the different documents and forms used in each of the phases.

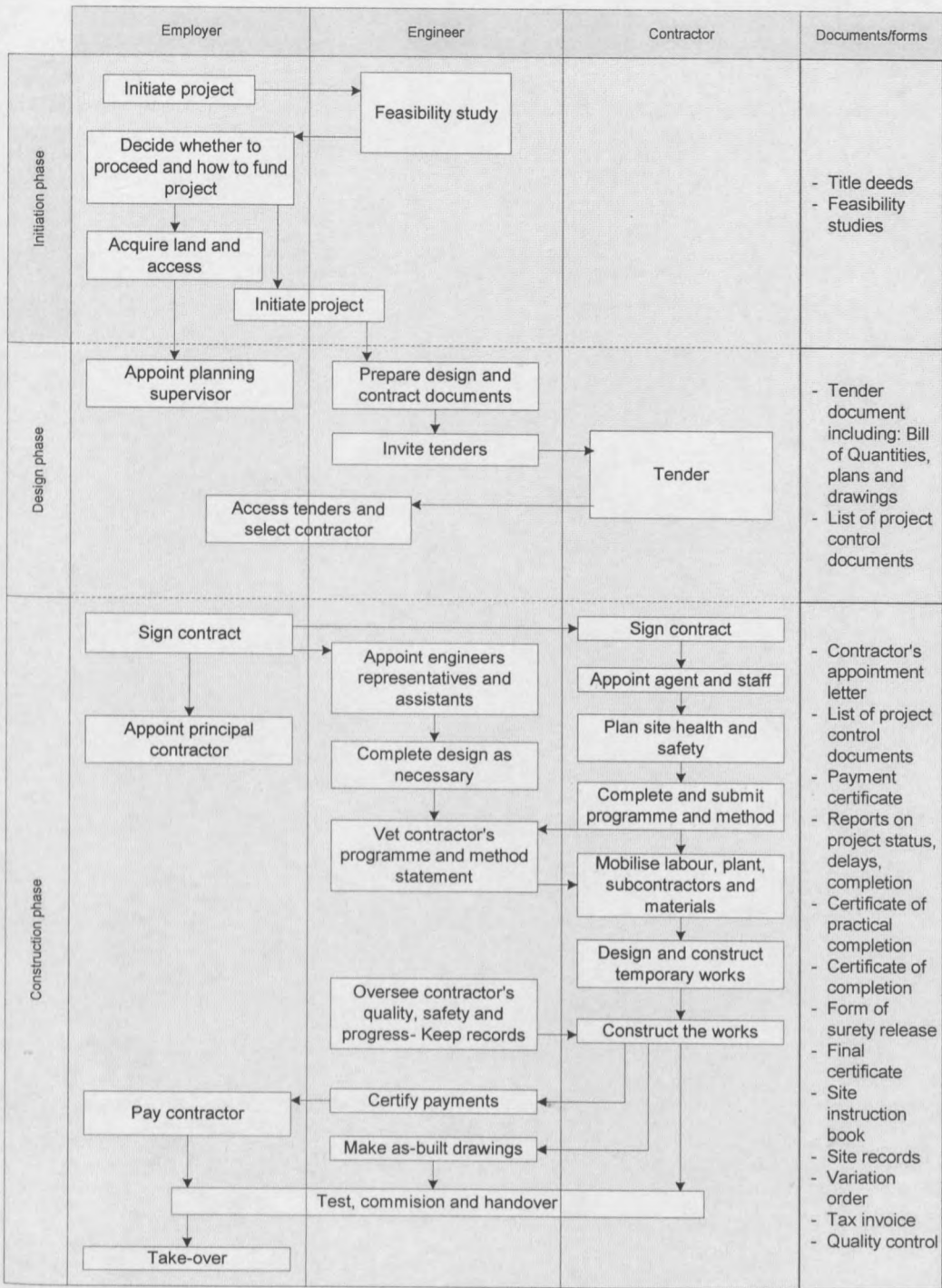


Figure 14 Project procedures and documents

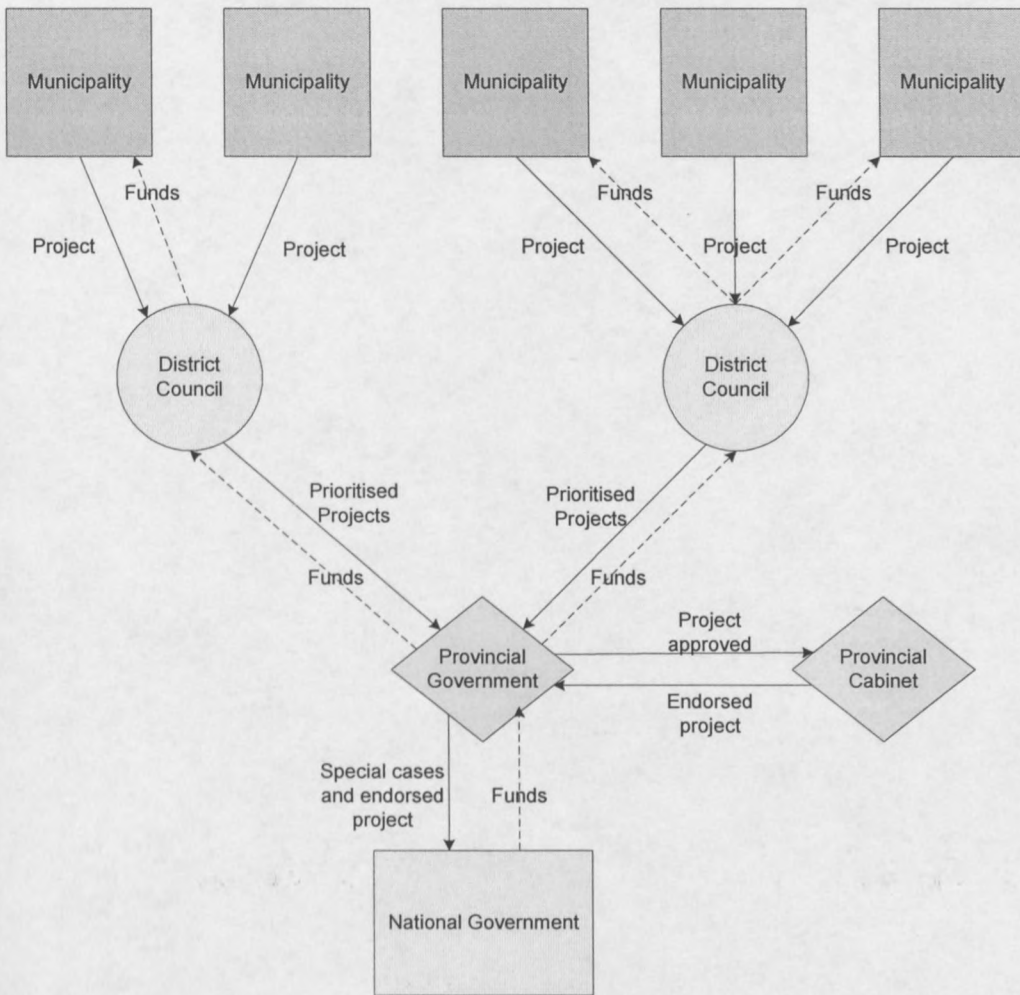
## **7.2 Project Information Flow in the Consolidated Municipal Infrastructure Programme**

Project information flow in the Government starts when a need is identified on local, provincial or national level. When a municipality identifies a need, a project to provide for this need is established and is sent to the district municipality. The district municipality will accumulate all projects from all the municipalities under its jurisdiction and will prioritise these projects. The relevant projects are then sent to the provincial project manager by all the different district municipalities. These projects are again prioritised and a shortlist of projects is drawn up, after which the projects are approved or rejected in principal. If a project is approved a business plan must be submitted by the municipality to the provincial project manager. On approval of the business plan, project recommendations are sent to the provincial cabinet and once these recommendations are approved, the provincial project manager will send the endorsement to the national project manager for approval of funds.

Claims for payment are sent to the provincial government by the district municipalities and the funds are then sent to the local municipalities who make the payments to the contractors.

Currently, the national government prescribes the use of the Perform Developer system, where data is entered into the system after the projects have been short-listed. This data is of a generic type, for example the name of the municipality, its location, the type of project and so forth. The national government also states that projects concerning water and sewerage must be accompanied by a technical report, which must be submitted to the Department of Water Affairs and Forestry for approval.

[Dieter Kopke, CMIP]



**Figure 15 Project information flow in Government**

### **7.3 Project Information Flow during Construction**

When the final designs for a project are complete, the project is ready to go out on tender (Figure 14). A tender document is provided to the contractors, which must be completed and handed back to the engineers before the closing data for the tender. The different tenders are then compared and the engineer, together with the employer, decides on which contractor to employ. To confirm the appointment of a contractor the engineer will send a letter of appointment to the contractor, confirming his employment.

After the contractor receives the letter of appointment, both the employer and the contractor will sign the contract, after which the works will commence.

The first claim will be submitted by the contractor to the engineer, usually towards the end of each month. The engineer will then certify the claim and send a payment certificate back to the contractor who will sign it and submit it to the client for payment. The payment to the contractor must then be made as specified in the tender document, usually within 30 days.

When the last payment certificate is submitted, the retention amount for the contract is reduced, usually by half, and is added to the payment certificate. At this stage, the engineer must point out all faults in the works and after all faults have been rectified, the engineer will issue a '*Certificate of Completion*' for the works. This is the start of the specified maintenance period. When this period lapses, the final inspection of the works is done and the remaining retention money is paid to the contractor. The engineer will then send a certificate for final completion of the works to the contractor, after which the contractor is finished with the works. At this stage control of the project is passed to the owner, typically a municipality, who will then become responsible for the management of the facility.

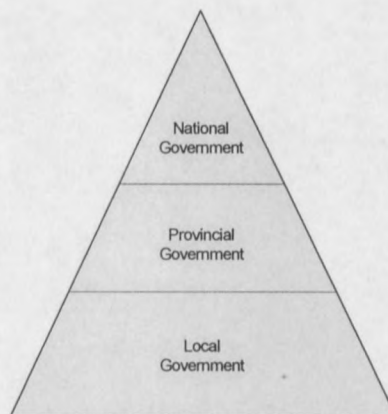
[J. van Zyl, Civil Engineer]



## CHAPTER 8

# ActionIT Organisation and Project Information Model Description

ActionIT is a non-profit standards development initiative aiming at creating broadly accepted interoperability standards for Decision Support Systems in the South African Government. The standards are intended to be used across all three spheres of the South African Government (Figure 16), i.e. local, provincial and national Government, but especially local government. It provides models for Project Management in Government, in particular the model 'Projects in Government'.



**Figure 16 Government Structure**

### 8.1 ActionIT structure

ActionIT consists of a Board of Directors and Technical Committees. The Board of Directors is responsible for steering the committees, for management and for all processes and legal aspects. Their responsibility further includes the final approval of

specifications produced by the Technology Committees, requests for proposals, external communications and ensuring that ActionIT meets the needs of its members. The committees on the other hand are responsible for steering the workgroups and for reporting to the Board of Directors. The responsibilities of the committees include:

- developing a research framework within which each of the workgroups can focus on a specific aspect of the specification.
- establishing appropriate workgroups and recruiting contributors from member organisations.
- co-ordinating and monitoring the progress of the workgroups.
- following the progress of other local and international standards development initiatives.
- Synthesising the ActionIT specifications from the contributions of the workgroups.

ActionIT subscribes to an open consortium based approach, much like the World Wide Web Consortium, to develop interoperability specifications that will facilitate the co-operative sharing of information. The objective of this approach is to achieve greater co-ordination of development and management processes in the Government.

[[www.actionit.org.za](http://www.actionit.org.za)]

## **8.2 Projects in Government Model (First version)**

One of the results produced by the ActionIT workgroups is a document titled “A Conceptual Model of Projects in Government”, which is available on the ActionIT website. The document describes a conceptual model for ‘*Projects in Government*’ comprising of: a ‘*System Model*’, a ‘*Process Model*’ and an ‘*Enterprise Data Model*’.

### 8.2.1 System Model

This model consists of system elements comprising of components and processes. The components define the static structure of the conceptual model and are as set out below. [Taken from the document 'A Conceptual Model of Projects in Government']

**Dependence** – Defines the dependence among ProjectElement components.

**Role** – Defines the role for a contract / played for a contract

**Party** – Defines an active contributor to a Contract – The terms Person and Group can be considered for use here. The Party/Person/Group can be (player of) active for a Contract/Appointment. Government organisational groupings of persons, in all three tiers of Government i.e. national, provincial and municipal (district and local) are catered for.

**Contract (or Appointment)** – Defined for a ProjectElement and can serve (support) a Role and have a Party/Group/Person active in (played by) the Contract.

**Location** – Defines physical location served by a ProjectElement and can be classified using Dimension.

**TimeRecord with specialisations TimePoint and TimePeriod** – Defines the timing of ProjectElement components and can be classified using Dimension.

**Deliverable** – Defines an outcome/output component of a Project, which can be grouped as accounts. This component also needs to be used for a Resource component, which is structurally and functionally equivalent.

**Entry** – Defines an Entry against a deliverable which is of type observation i.e. can be quantitative, qualitative or descriptive.

**Transaction** – Linking components of type Entry to component Transaction allows financial control in an accounting sense.

**Dimension** – Defines a classification logic for components such as Location, TimeRecord and ProjectElement. ProjectElements can be discrete (start and terminate) or continuous.

**Stereotype conceptual model entities:** Stereotypes are extensions of the standard modelling (UML) vocabulary derived from the standard vocabulary, which are specific to this model. Composite, Account, Observation, and Transaction are defined for this model and are described later.

**Stereotype mappings:** Hierarchy allows collection of components as parents and children and sequence indicates an ordered collection of components allowing duplicates.

### **8.2.2 Process Model**

The Process Model focuses on project monitoring and assumes that projects have a hierarchal structure. It provides for behavioural modelling, supports concepts such as risk management and control via double entry bookkeeping and allows sub processes such as financial, social, legal and environmental aspects.

The dataflow for this model consists of project sub-processes and includes:

- Formulation
- Registration
- Evaluation
- Prioritisation
- Approval
- Monitoring
- Closure

[For more detail see '*A Conceptual Model of Projects in Government*', available at [www.actionit.org](http://www.actionit.org).]

### **8.2.3 Enterprise Data Model**

The Enterprise data model focuses on being a generic unifying conceptual object model for projects in government. It makes provision for object, version, configuration and context management. It further defines stereotypes (otherwise known as entities) with associations, which promote compactness. The entities defined by the model are:

**Composite** – A group of projects consisting of one or more deliverables and zero or more projects.

**Account** – A group of deliverables of which the values can be increased or decreased through a series of entries.

**Observation** – This is an entry against a deliverable and can be quantitative, qualitative or descriptive if it is classified as an observation.

**Transaction** – Linking entries to a transaction allows financial and inventory control in a system of pseudo-accounting.

The model layout is shown in Figure 17. [Taken from the document '*A Conceptual Model of Projects in Government*']

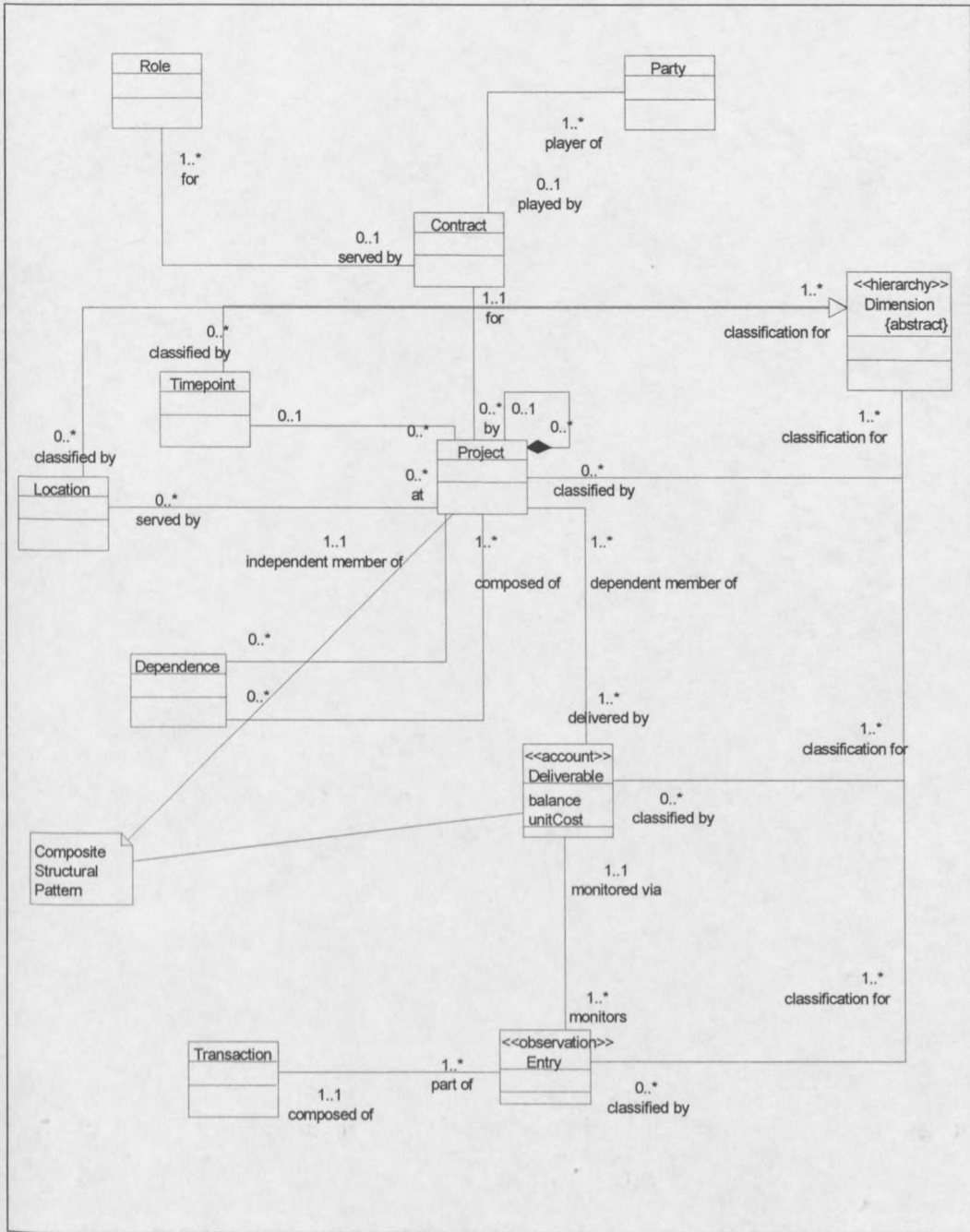


Figure 17 The ActionIT Enterprise Data Model

The model further provides for mapping between these types and when associations of multiplicity of one and higher is encountered, the resulting collections are dealt

with either as a hierarchy, treating the collections as parents and children, or as a sequence.

Another feature supported by the model is aggregation. This is an important feature as it is used in the decision making process where initiatives are measured by aggregating the projects, describing the strategy or initiative. Aggregations provided for are spatial, temporal, observation and composite aggregations.

An example of the *Enterprise Data Model* would be to consider a project involving the development of low-cost housing. Typical accounts for this project would be as shown in Figure 18.

Reference	Deliverable	Unit	Value per unit	Dimension Functional Classification	Balance
s19	Sites of size 200 sq m	site	R4000.00/site	Technical	1000 000
s23	Subsidy in R12500 band	subsidy	R12500.00	Financial Funding	2500 000
s17	Surveying	h	R/h	ProfessionalFee	300
s25	Title Deeds	Deeds	R300.00/deed	Legal Documentation	175
s26	Town planning fee	h	R/h	ProfessionalFee	500
s21	Storm water	m	R800.00/m	Technical Services	2500
s22	Streetlights	lights	R1200.00/light	Technical Services	785

**Figure 18 Example of accounts for the Enterprise Data Model**

## CHAPTER 9

### **Description of Implemented Applications and System**

Following the research and study of the previous chapters, this chapter describes the implemented applications and system that was designed. It starts with a description of the user and application/system requirements and then continues to describe the structure and processes of applications and system.

#### **9.1 User Requirements**

The users of these applications and system are municipalities and government or governmental organisations having to convey project data between them. The data to be conveyed includes project forms, reports and any other project data. To enable this communication, municipalities need to be able to:

- Supply information regarding projects to the Government.
- Perform the information supplying process faster than with traditional methods.

The Government needs to be able to:

- Retrieve all information supplied by municipalities.
- Confirm reception of data.
- Store the data.
- Aggregate data from different projects.
- Process and transform the data for report purposes.
- Generate reports from the data.
- Interpret the reports.
- Give feedback to the municipalities.



## **9.2 Application and System Requirements**

Requirements for the applications and system were defined by the user's needs and were important for design as it dictated the layout and functionality of the applications and system. What follows is a description of the requirements.

### **9.2.1 Information**

The first requirement was the gathering of information relevant to project management. Functionality must be provided for easy and efficient data entry by using a graphical user interface. Users must also be able to update their information or to submit new information as a project continues.

### **9.2.2 Time Efficiency**

Time efficiency is required, with the aim being to gather and process information within a much shorter time than the systems currently implemented by CMIP. Processes like report generation, aggregation of projects, data entry and storage of data must be accomplished quickly and efficiently.

### **9.2.3 Data storage**

Data storage is an important requirement for project information management. The storage and retrieval of data as well as the dynamic creation of unique database tables must be enabled.

### **9.2.4 Aggregation**

Aggregation of project information is required in order to enable data for different projects to be combined for reporting purposes. Functionality should be provided to aggregate any amount of projects.

### **9.2.5 Reports**

The generation of reports is a fundamental requirement for project management. The system should include functionality to generate different types of reports as well as visualisation of reports.

## **9.3 Applications and System Designs**

As an attempt to incorporate and utilise the relevant technologies into the AtionIT Enterprise Data Model, using CMIP as an example and to provide for all the user requirements, the applications and system described below were designed (Figure 19).

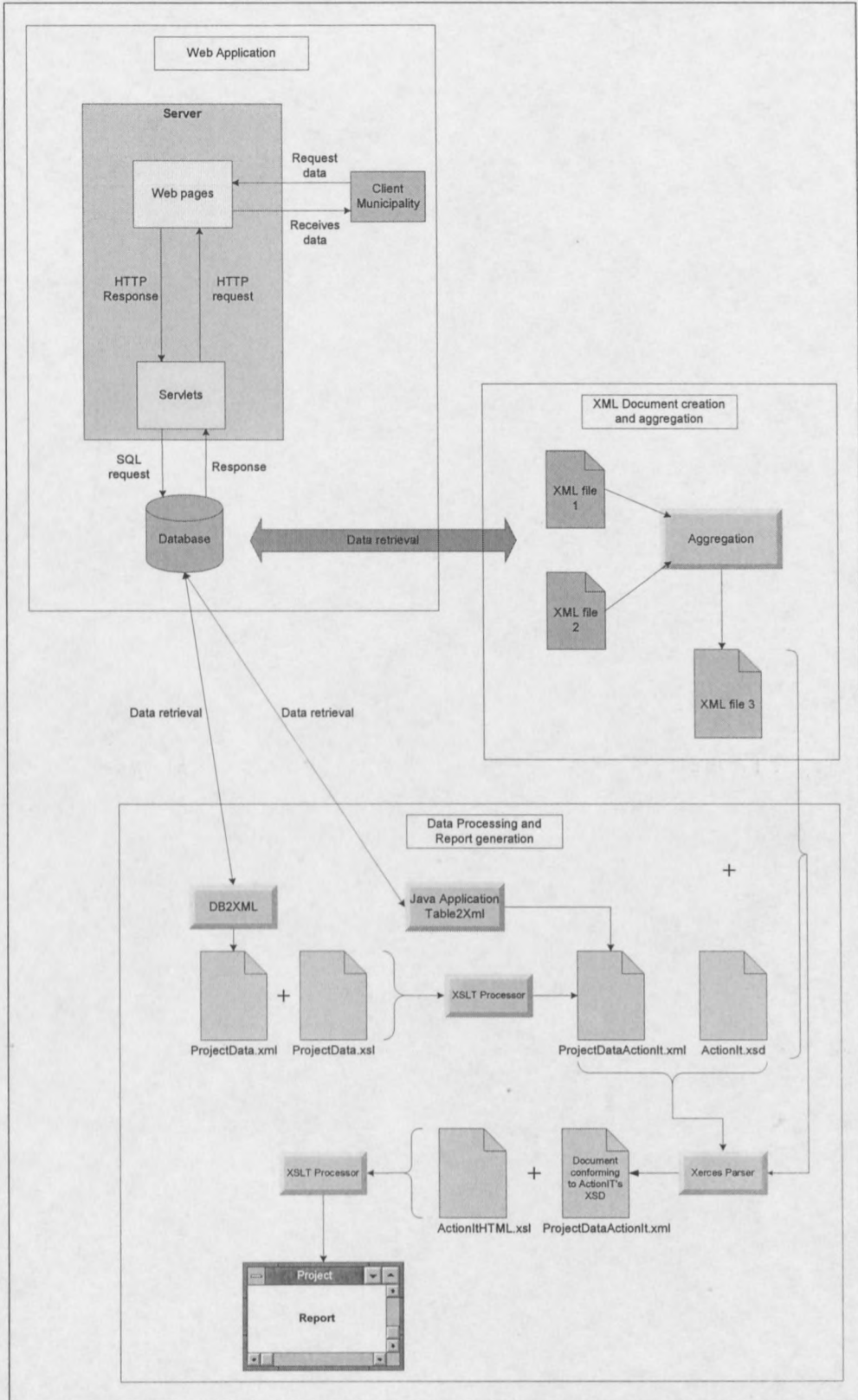


Figure 19 System diagram

**1. Web Application:**

The application provides functionality for online completion of project information forms and the storage and management of the data.

**2. XML Document Creation and Aggregation Application**

The application was designed for two reasons: to convert data from the database to XML using a Java application and to provide functionality for aggregating two or more XML documents containing project data.

**3. Data processing and Report generation System:**

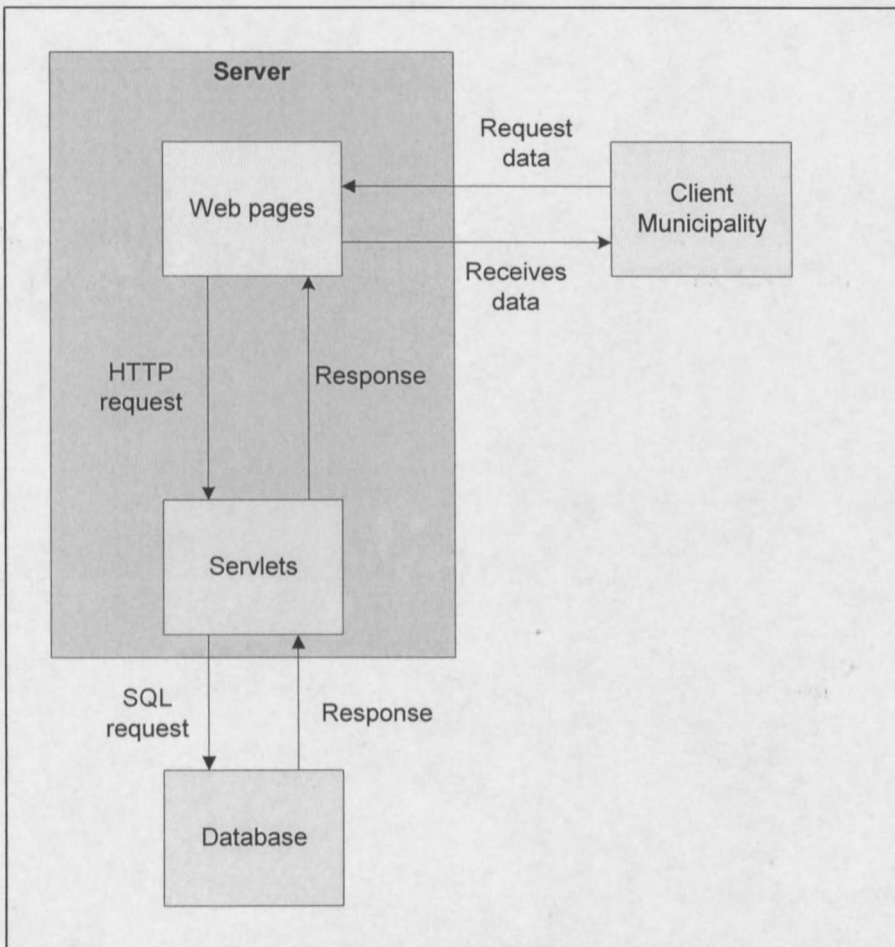
The Data processing system enables the conversion of data from a database to XML; the transformation of this document into another XML document with a different structure adhering to prescribed specifications; the validation of this document and a final transformation of the document to create a report.

## 9.4 Web Application

The Web Application's components, structure and functionality are described below.

### 9.4.1 Components and Structure

The application structure is as shown in Figure 20 and consists of clients communicating with a server through web pages, servlets, a database being queried by servlets using standard SQL and clients requesting information (See Appendix C for setting up and installing the application).



**Figure 20 Application Structure**

### **9.4.1.1 Web pages**

Web pages are used for data entry. Pages contain project management forms and can be completed online. The layouts of the pages were obtained by analysing the CMIP implemented system forms with special attention to the information being conveyed between CMIP, municipalities and the Government. It was found that a variety of different forms are in use for project data distribution and as part of the web application, these forms were incorporated in the design of the web pages. The forms can be selected on the website home page by clicking on the links. The web pages were created using HTML and the provided POST method is used to send a HTTP request over the Internet to the server. The request is sent when a user presses the 'Send' button at the end of a form and includes the servlet name responsible for handling the data as well as all the data filled in on the form. Each field that can be filled in on the form has been given a field number, which is used as reference for the field. This reference is used for mapping of the fields to universal account names.

Access to the web pages is controlled by giving each municipality a username and password, which is required before anything can be submitted or updated. A municipality only has access to their specific data and therefore is not able to tamper with other information in the database.

The main advantage of the web application is that it allows the online entry of data, thus making the gathering and storing of data much faster than with traditional 'paper based' systems. This is due to the fact that data only needs to be entered once into the application and not by all the parties involved. It provides for automatic and instant confirmation whether data was successfully submitted.

### **9.4.1.2 Server**

The server used for this application was the Apache Tomcat 4.0 web server, available for downloading from <http://jakarta.apache.org>. This was decided based on the fact that it is available as free source and supports servlets, therefore suiting this application very well.

### **9.4.1.3 Servlets**

In this application servlets are used for:

#### **1. Password based access control**

Servlets are used to verify a client's integrity by comparing the username and password given by a client. If the username and password match those in the database, the client is allowed to submit or retrieve data.

#### **2. Handling requests from the web pages**

For each different form that can be completed on the web pages, a separate servlet was created and registered on the server. When the server receives a POST request from the web pages, the request is sent to the servlet specified in the request. Upon receiving the information the servlet executes and starts reading all the information sent in the request.

#### **3. Connecting to the database**

Once the servlet finishes reading the data from the request, it continues by specifying the JDBC-ODBC bridge driver for connecting to the database, in this case the *sun.jdbc.odbc.JdbcOdbcDriver*, and then connects to the database specified in the URL. For instance, if the name of a database to be used is DB1, the URL value would be 'url = jdbc:odbc:DB1'.

#### **4. Creating tables**

After a connection to the database is established, the servlet creates a table to store the data. As the information of each submitted form is stored in a table with a

unique name, the servlet first verifies that the table does not yet exist before creating it. If a table is already in existence, it means that a form is being updated.

## **5. Mapping and storing of data**

With a table in place, the servlet starts reading the data into the table. The table created contains three columns: Ref, ID and DATA. The Ref column is filled with the field reference number read from the request. The servlet then uses the reference number to run through the list of universal accounts contained in the Mapping2 table to find the corresponding account name. This is filled into the ID column. The corresponding data read from the request is then filled into the DATA column.

## **6. Retrieval of data from the database**

Once the data is available in the database, it can be accessed and retrieved by the servlets. Here servlets are only used to verify usernames and passwords, but it can also be used to retrieve and display information, for instance if a municipality wants to view all the data it has submitted.

## **7. Constructing responses**

The last aspect handled by the servlets is responses. This is done by dynamically creating a HTML web page that is sent to the client and displayed using the client's browser. This is used for confirming that data has successfully been submitted and for informing a user to retype a username and password if the wrong values were entered.



#### 9.4.1.4 Database

The database used for the application was the Microsoft Access 2000 database, which could be queried through standard SQL statements.

The database consists of the tables described below:

1. **ApplicationTables**, containing all the names of the existing xxxxApplicationData tables, where xxxx is a unique identification number for a municipality. This table is used to check if municipalities have previously submitted data. A similar table, **LevelOfServiceTables**, exists for the Level of Service Questionnaire form.

Table
Default
987ApplicationData
88976ApplicationData
854ApplicationData
35ApplicationData
2243ApplicationData
2232ApplicationData

**Figure 21** *ApplicationTables* table

2. **xxxxApplicationData**, tables created dynamically and used to store application data submitted by municipality. The table contain three columns: **Ref** – a reference to the universal account/deliverable name, **ID** – the universal account/deliverable name, **DATA** – the data read from the form corresponding to the account/deliverable name.

Ref	ID	DATA
f1	Last Modified	12/3/2000
f2	Application Receive	12/5/2002
f3	Application Acknowledge	15/7/2002
f4	Project Name	Building houses
f5	PFD Reference number	10023
f6	National Reference n	100005
f7	Applicant	Drakenstein Municipality
f8	Applicant title	Mnr
f9	Applicant Surname	Smith

Figure 22 xxxxApplicationData table

3. **Passwords**, a table containing the usernames and passwords for all the users.

This table is used to check a user's username against the supplied password.

Username	password
Durbanville	durb
Klawer	klaw
Pinelands	pine
Stellenbosch	stel
Vredendal	vred

Figure 23 Passwords table

4. **Mapping2**, a table containing data for mapping each field being read from a form to an universal account/deliverable name. The table contains two columns: **Field** – supplying the field number for reference purposes and **AccountName** – containing the universal account/deliverable names corresponding to the field.

Field	AccountName
f1	Last Modified
f2	Application Receive Date
f3	Application Acknowledge Date
f4	Project Name
f5	PFD Reference number
f6	National Reference number
f7	Applicant

Figure 24 Mapping2 table

5. **xxxxAccountRegister**, a table containing the accounts of a specific project. The table consist of six columns:

Column 1: **Reference** – a reference number to a universal account/deliverable name.

Column 2: **Deliverable** – the name of the account.

Column 3: **Unit** – the units in which the account is increased or decreased.

Column 4: **Value per unit** – the value of one unit.

Column 5: **Dimension** – the classification of the account.

Column 6: **Balance** – the available units with which the accounts can be increased or decreased.

Reference	Deliverable	Unit	Value per unit	Dimension	Balance
f40	Access Roads	km	50000	Technical Feasibility	100
f41	Bridging Finance	R	1	Financial Funding	1000000
f42	Bulk Electricity Supply	kw	75	Technical Feasibility	2500
f43	Bulk Sewer	MI	15	Technical Feasibility	1005
f44	Bulk Storm Water	MI	8	Technical Feasibility	2100
f45	Bulk Water Supply	MI	100	Technical Feasibility	4800

Figure 25 xxxxAccountsRegister table

These tables are used for managing a project's data and can be aggregated for reporting purposes. (See Appendix C for installation procedures.)

## **9.4.2 Application Functionality**

The functionality for the application is as described below. It explains how the requirements for the application were met and what technologies were used.

### **9.4.2.1 Online Data Entry**

Currently the process of project data accumulation from municipalities is a long and inefficient process as it is still partly paper based. This causes the duplication of data (data must be filled in on forms and later be entered into computer programmes) and a longer waiting time to receive data as the forms are delivered physically. To streamline the process of data accumulation, a web application has been designed where municipalities can complete all the necessary forms and submit the data online. Municipalities are also enabled to update their information or to enquire about the status of an application.

### **9.4.2.2 Logging of events**

Although not implemented in this application, the logging of events can be incorporated in the application. It will typically be implemented to store the time, the date, the user and what data has been changed or entered when any modifications to the database are made. This can be used for later reference or for security reasons.

### **9.4.2.3 Data storage**

An important feature of the application is the storage of data and is as explained in section [9.4.1.3 Servlets](#)

### **9.4.2.4 User feedback**

The application includes functionality to create feedback for clients. The servlets are responsible for the creation of dynamic web pages, based on the client's action, which are sent back to the client as a response.

## 9.5 XML Document Creation and Aggregation

### Application

This application enables the aggregation of two or more project data sets either in XML format or in database tables. These data sets must conform to the ActionIT Enterprise Data Model. It further enables the transformation of data from a database to XML documents.

#### 9.5.1 Components and Structure

The structure of this application is as shown in Figure 26 and consists of Java classes, which interact with each other in order to aggregate projects and to create XML documents.

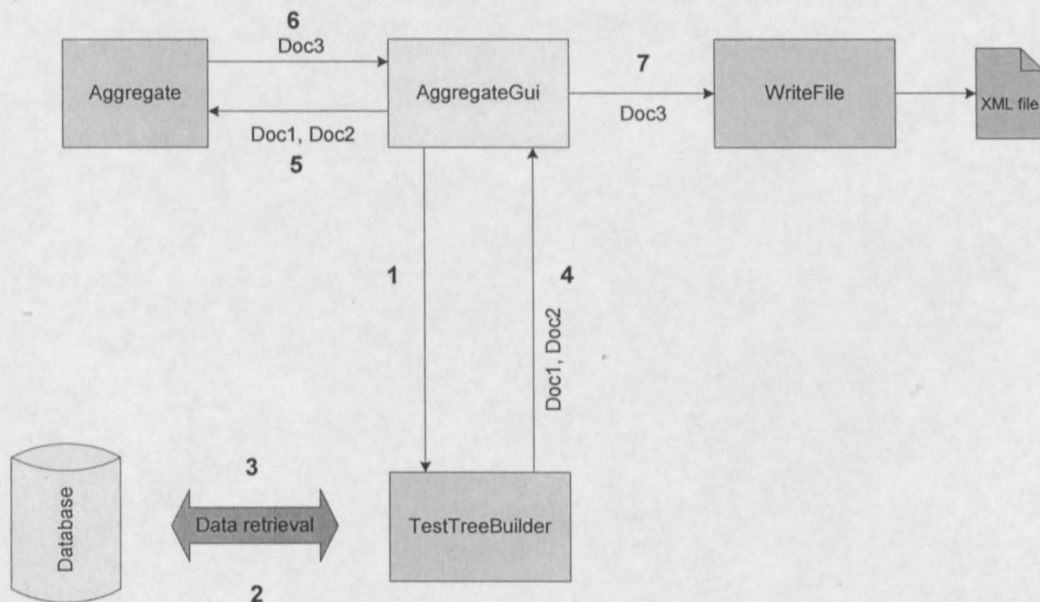


Figure 26 Aggregation application classes and data flow

### 9.5.1.1 AggregateGui class

The AggregateGui class is responsible for the creation of a graphical user interface (GUI) as shown in

Figure 27, as well as for the instantiation of the objects needed for aggregation. The GUI provides the user with a choice either to specify two tables or two XML documents to be aggregated. These must conform to the ActionIT Enterprise Data Model. When the ‘Aggregate XML files’ button is pressed, the two XML files specified are aggregated and the resulting document, as well as the original two documents, are displayed in the windows. The resulting document will also be saved to the directories specified. The same will happen when the ‘Table2XML’ button is pressed. Objects for aggregation are instantiated in sequence as shown in Figure 26, the numbers indicating the sequence.

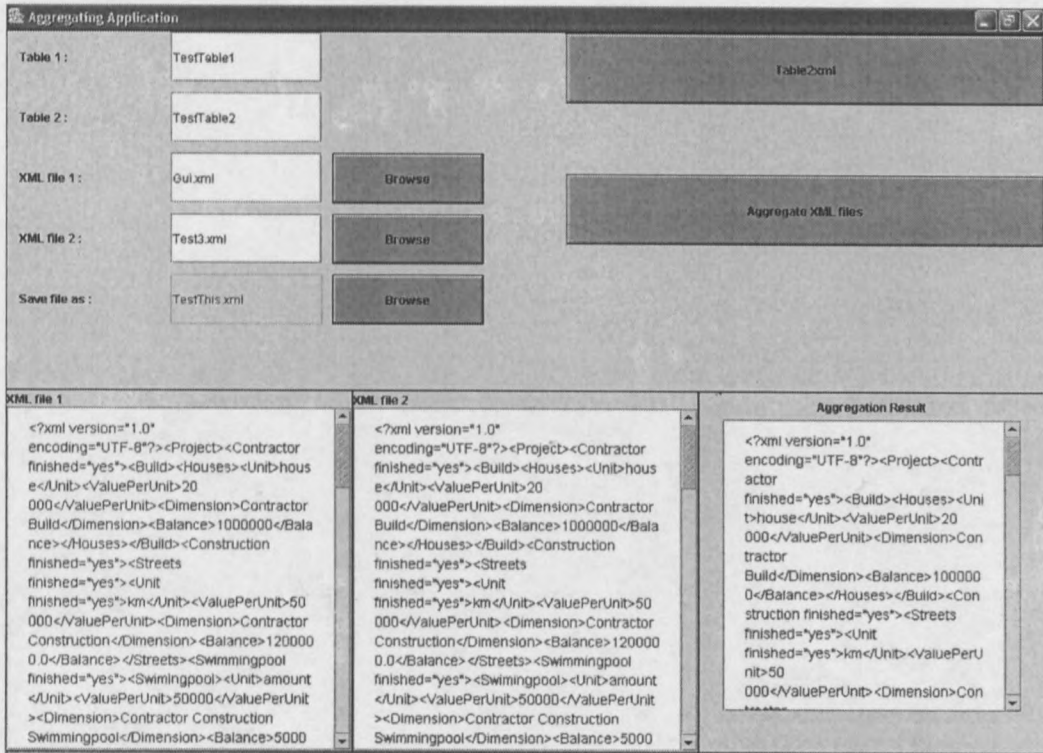


Figure 27 AggregateGui graphical user interface

### **9.5.1.2 TestTreeBuilder class**

This class takes a database table name as input, which it uses to query the database. It then retrieves the information from the specified table and creates a JDOM document of the data conforming to the ActionIT Enterprise Data Model. It then returns this document to AggregateGui.

### **9.5.1.3 Aggregate class**

To perform aggregation the JDOM documents created by the TestTreeBuilder class are handed to the Aggregate class and aggregation is then performed. This is done using JDOM and methods of aggregation, described in the sections below. When the aggregation process is complete the Aggregate object returns the aggregated document to AggregateGui.

#### **9.5.1.3.1 Implemented classes**

For the aggregation of projects it was decided to make use of tree structures. This decision was based on the ease of aggregation of two or more trees and the easy output of data in comparison to other methods. For this purpose the Java Document Object Model (JDOM) classes, created by Brett McLaughlin and Jason Hunter, were used.

#### **9.5.1.3.2 Method of Aggregation:**

The first step towards project aggregation is accomplished by writing the data of each project into a JDOM document representing a tree structure, which can be manipulated for aggregation purpose. The JDOM document/tree objects is constructed using the JDOM construct:



Document doc = new Document(new Element (“root”));

with “root” being the root element of the tree. For aggregation two trees are handled at a time, mapping the one onto the other (Figure 28). Mapping of the trees is accomplished by comparing the nodes in the first tree with those in the second, mapping existing nodes onto each other and adding nodes existing only in the second tree to the first. After this is done the document can be outputted as a XML file containing the aggregated data, ready for report generation and use by clients.

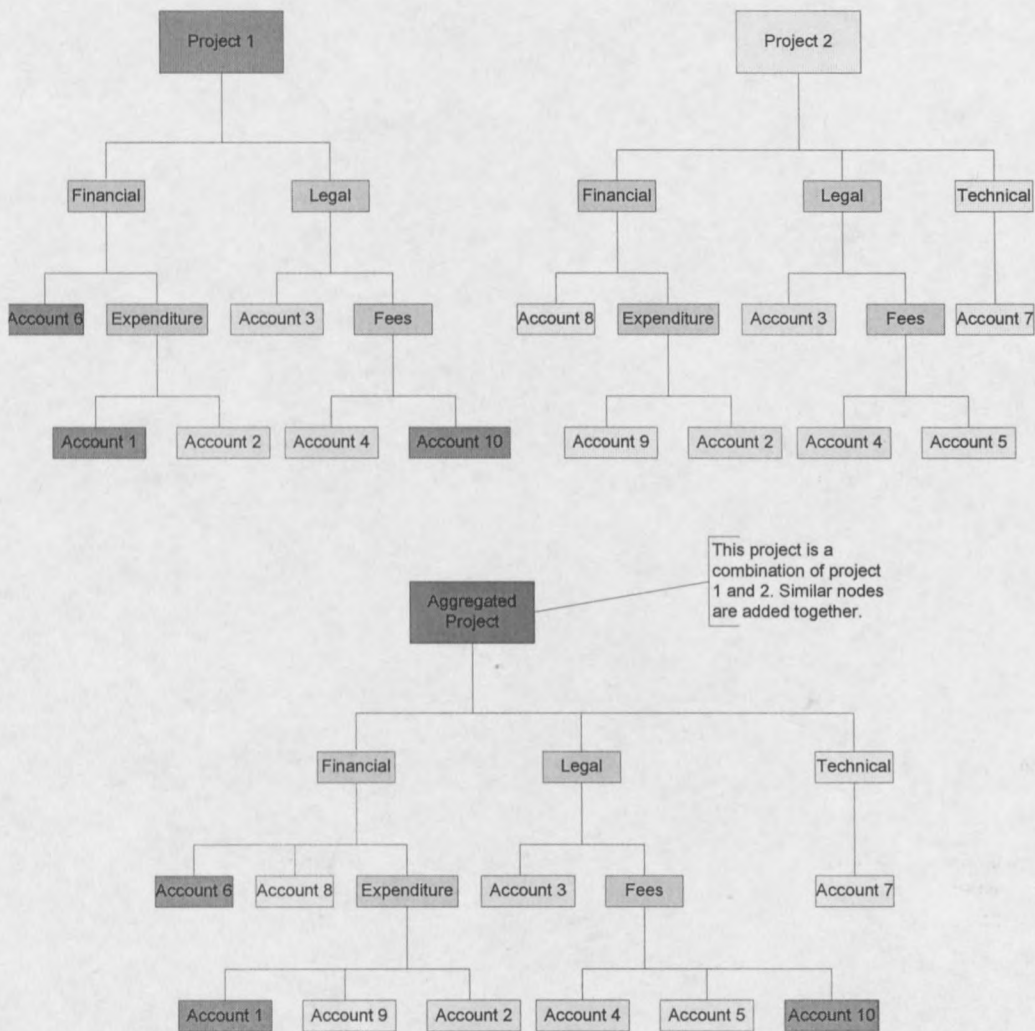


Figure 28 Project Aggregation

### 9.5.1.3.3 Server and Client side Aggregation

Aggregation of project data can be performed either on the server or client side. It is however advantageous to enable aggregation on both server and client sides. This is profitable as it provides for a client to receive XML files that were created on the server side from two or more projects and to aggregate this with other similar files received from other servers. This will also give a client the option as to where aggregation must be performed. (Figure 29)

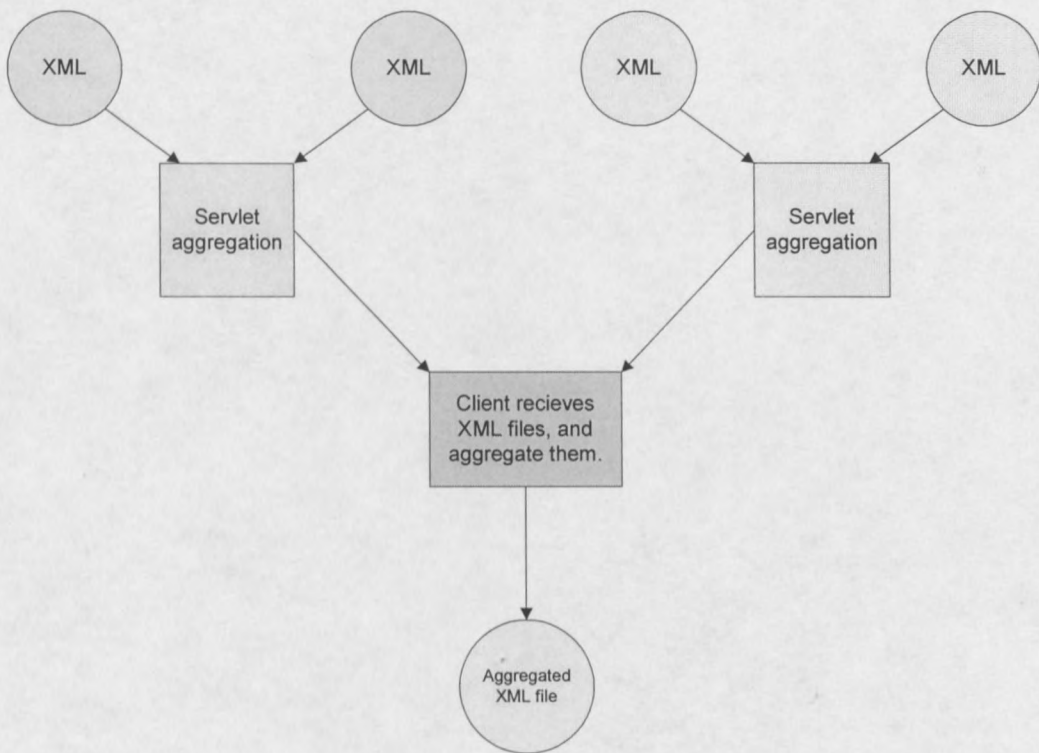


Figure 29 Aggregation of multiple XML documents

#### **9.5.1.4 WriteFile class**

This class is responsible for converting the JDOM document that was aggregated to a XML document. After the transformation it writes the file to the directory specified by the user.

### **9.5.2 Application Functionality**

The functionality of this application is as set out below and consists of the creation of XML documents and the aggregation of two or more of these documents.

#### **9.5.2.1 XML Document creation**

The application provides functionality for the conversion of data from a database to XML by executing a Java application. The application retrieves data from tables specified and creates XML documents with a structure conforming to the ActionIT Enterprise data model (Version one).

#### **9.5.2.2 Aggregation**

Functionality for project aggregation is provided. It enables the aggregation of multiple XML documents containing project data structured according to the ActionIT Enterprise Data Model. It then gives a XML file containing the aggregated data as output.

#### **9.5.2.3 Graphical User Interface**

A Graphical User Interface is provided as shown in Figure 27. It provides functionality for the user to:

- Specify the tables to be aggregated or;
- the XML documents to be aggregated.
- Choose the files from a display when clicking a browse button.
- Specify where to save the resulting file.
- View the input files as well as the resulting file.

## **9.6 Data Processing and Report Generation System**

This section describes the implemented system which generates reports using XML and related technologies.

### **9.6.1 Components, Structure and Functionality**

The system structure is as shown in Figure 30 and consists of six components; importing data to the database, converting the data to XML using DB2XML or a Java application, creating stylesheets for converting the XML documents to adhere to a specific structure, processing the XML document using a XSLT processor, validation of the XML document and conversion of the XML document to HTML format.

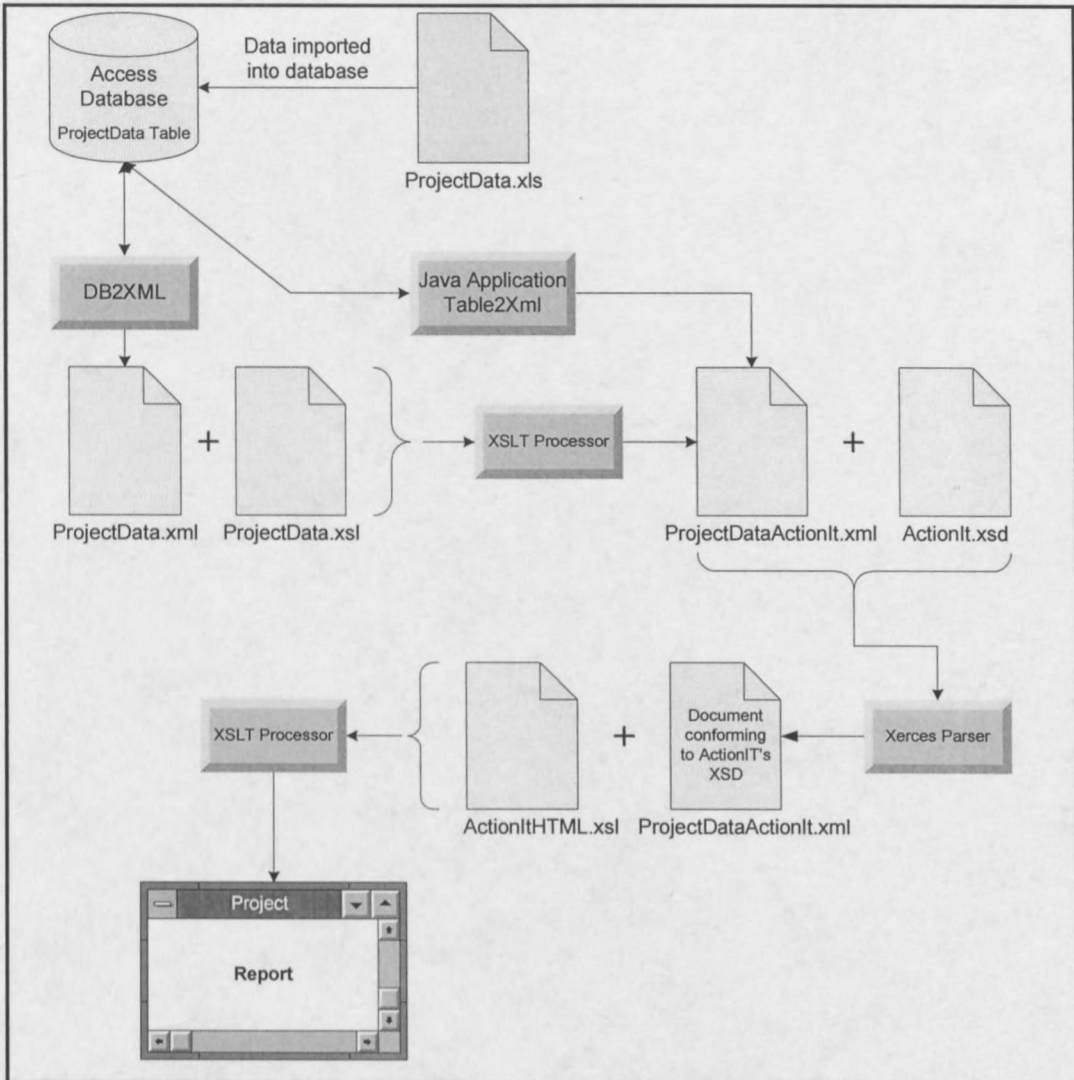


Figure 30 Data Processing

1. Obtaining project data and storing it in the database

Project data can be obtained from any source and in any format as long as it can be imported into the database. For this system Microsoft Access, using the software's provided tools, was used. The data obtained must be stored as a table in the database and must be available for use.

## **2. Converting data from the database to XML**

Conversion of data from the database to XML was done in two ways. The first method was by using DB2XML, a freeware application allowing the specification of a database to use, the DTD to apply if any, the table to convert, the tags to use, the output destination and more (See Appendix B).

The second method was by using Java and JDOM as described in section 9.5.2.1 XML Document creation.

## **3. Creation of a XSL stylesheet**

A XSL stylesheet for the transformation of the XML document was programmed using CookTop2.200. This is a XML editor that is available as freeware (see Appendix A). The stylesheet was specific to the provided project information and was designed to change the structure of the XML document to conform to the structure specified in the ActionIT XSD for project information. This stylesheet is however just needed for the XML file created using DB2XML, as the Java application was designed to build the XML file to conform to the ActionIT XSD without needing transformation to adhere to the specified document structure.

## **4. XSLT processing using the XML document and XSL stylesheet**

The transformation of the XML document created with DB2XML was done using the Apache Xalan XSLT processor, which can be found at <http://xml.apache.org>. It is also distributed as part of CookTop2.200, which was used to process the document as it provides a user interface for XSLT processing. A XML document and XSL stylesheet need to be provided.

## **5. Validating the resulting XML document against the provided XSD schema**

For validating the XML document the Xerces2.2.0 validating parser was used in combination with Java. This was done to ensure that the transformation of the XML document was performed correctly and that the XML document structure was the same as that specified by the XSD schema, enabling the application programme to use the XML document for report generation.

## **6. Report generation and visualisation**

Reports were generated from the converted XML document after it was validated. This was done by first creating another XSL stylesheet. The stylesheet and the validated XML document were again handed to the XSLT processor, which converted it to a HTML document. This document could then be displayed in a browser.



## CHAPTER 10

### Examples of Implemented Systems and Applications

This chapter provides three implementations/examples of the applications and systems as described in Chapter 9. The first example deals with the gathering and storing of information from municipalities, the second with project aggregation and the third with report generation using XML and related technologies.

#### 10.1 Web application Example

Data for this system was gathered by a web application comprising of a home page with options to select interactive forms. Two forms have been included for demonstration purposes: an Application form and the Level of Service Questionnaire form. For this example only the Application form has been filled with data to provide data for further demonstrations of the system. Figure 31 and Figure 32 show the home page and an extract from the Application form.

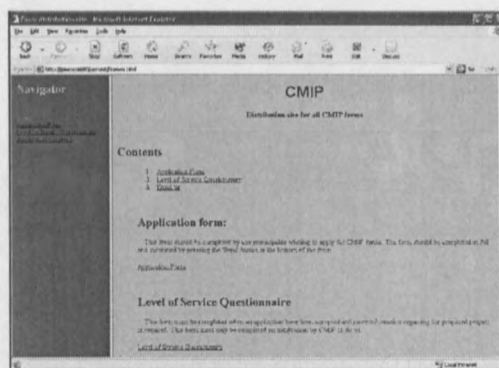


Figure 31 Website homepage

Consolidated Municipal Infrastructure Programme  
Application Form CMIP 1

Municipality:  Project Ref No:

Project:

Last Modified:

ALL FIELDS TO BE COMPLETED IN FULL

Registration:  Date Application received:  City:  Administration:

Project Name:  Existing business:

PPD Ref No:

National Ref No:

1. APPLICANT DETAILS

1.1 Name of Contact Person Responsible for Document:

**Figure 32 Application form provided as a web page**

After the form was completed it was submitted by pressing the ‘Send’ button at the bottom of the form. Upon successful completion of the process the user received a confirmation message as shown in

Figure 33.

Application completed successfully

Please allow 10 working days for processing of applications

Thank you and good bye

**Figure 33 Response after successfully submitting data**

When a form is submitted, the server passes the request to the corresponding servlet as described in section [9.4.1.3 Servlets](#) and all the data is sent to the Access database and stored in a table. The table is specific to the municipality submitting the data. Figure 34 shows a table created for a municipality and the data stored.

Ref	ID	DATA
field1	Last Modified	12/3/00
field2	Application Receive	12/5/02
field3	Application Acknowled	15/7/04
field4	Project Name	Building houses
field5	PPD Reference number	10079
field6	National Reference n	100005
field7	Applicant	Draekendin Municipa
field8	Applicant title	Mer
field9	Applicant Surname	Smith
field10	Applicant initial	J
field11	Applicant Position	Engineer
field12	Physical Address	Pearl, Longstreet 17
field13	Postal Address	Po Box 3324, Pearl
field14	Physical Address Pos	9083
field15	Physical Address Erna	9083
field16	Physical Address Cal	smith@email.maf
field17	Physical Address COD	021 44333454
field18	Postal Address Postal	083 33445544
field19	Postal Address Fax	021 7483334
field20	Postal Address Tel(O	98009
field21	Type of Applicant	Municipality
field22	Provincial Ref. Number	203
field23	Estimated Cost	6 200 000
field24	Project Start date	15/04/02
field25	Project Completion d	15/04/04
field26	Project Duration	112
field27	Project description	upgrading
field28	Project Motivation	no special case

Figure 34 Table created and stored in the database for a given municipality

## 10.2 Project Aggregation Example

To demonstrate project aggregation two files containing project data and conforming to the Enterprise Data Model were created and used for demonstration purposes as shown in Figure 35 and Figure 36. In general the data from the previous example, Web application, should be transformed to this format. For this step however the data transformation step was omitted, as all the project detail is needed, including the Bill of Quantities, to supply the opening balances, units, value per unit and so forth. Currently the web application does not provide support for a Bill of Quantities and for this reason already formatted data was supplied. It is however an important step for the system as a whole.

Ref	Deliverable	Unit	Value per unit	Dimension	Balance
f111	Houses	house	20 000	Contractor Build	1000000
f112	Streets	km	50 000	Contractor Construction	500000
f113	Lights	amount	20	Electrical Lights	20000
f114	Wiring	m	5	Electrical Lights Wiring	10000

Figure 35 Project data table 1

Ref	Deliverable	Unit	Value per unit	Dimension	Balance
f115	Swimmingpool	amount	50000	Contractor Construction Swimmingpool	500000
f112	Streets	km	60000	Contractor Construction	700000
f113	Lights	amount	30	Electrical Lights	25000
f114	Wiring	m	8	Electrical Lights Wiring	14000
f116	Connections	amount	40	Electrical Lights Wiring Connect	45000

Figure 36 Project data table 2

After these two tables were created it was aggregated by executing the Java application 'AggregateGui' and using the Graphical User Interface. The resulting file was saved as a XML document and is as shown in Figure 37.

```
<?xml version="1.0" encoding="UTF-8"?>
<Project>
  <Contractor finished="yes">
    <Build>
      <Houses>
        <Unit>house</Unit>
        <ValuePerUnit>20 000</ValuePerUnit>
        <Dimension>Contractor Build</Dimension>
        <Balance>1000000</Balance>
      </Houses>
    </Build>
    <Construction finished="yes">
      <Streets finished="yes">
        <Unit finished="yes">km</Unit>
        <ValuePerUnit>50 000</ValuePerUnit>
        <Dimension>Contractor Construction</Dimension>
        <Balance>1200000.0</Balance>
      </Streets>
      <Swimmingpool finished="yes">
        <Swimingpool>
          <Unit>amount</Unit>
          <ValuePerUnit>50000</ValuePerUnit>
          <Dimension>Contractor Construction Swimmingpool</Dimension>
          <Balance>500000</Balance>
        </Swimingpool>
      </Swimmingpool>
    </Construction>
  </Contractor>
</Project>
```

</Contractor>

<Electrical finished="yes">

<Lights finished="yes">

<Lights>

<Unit>amount</Unit>

<ValuePerUnit>20</ValuePerUnit>

<Dimension>Electrical Lights</Dimension>

<Balance>20000</Balance>

</Lights>

<Wiring finished="yes">

<Wiring finished="yes">

<Unit finished="yes">m</Unit>

<ValuePerUnit>5</ValuePerUnit>

<Dimension>Electrical Lights Wiring</Dimension>

<Balance>24000.0</Balance>

</Wiring>

<Connect finished="yes">

<Connections>

<Unit>amount</Unit>

<ValuePerUnit>40</ValuePerUnit>

<Dimension>Electrical Lights Wiring Connect</Dimension>

<Balance>45000</Balance>

</Connections>

</Connect>

</Wiring>

<Lights finished="yes">

<Unit>amount</Unit>

```
<ValuePerUnit>30</ValuePerUnit>  
<Dimension>Electrical Lights</Dimension>  
<Balance>25000</Balance>  
</Lights>  
  
</Lights>  
</Electrical>  
</Project>
```

**Figure 37 Aggregated project XML document**

After aggregation the resulting XML document is available for reporting purposes or for further aggregation with another XML document.

### 10.3 Data Processing and Report Generation Example

The data used to demonstrate this example is a Bill of Quantities obtained in Microsoft Excel format from the engineering firm Entech. An extract from the document is shown in Figure 38.

Item	Payment	Description	Unit	Quantity
		SECTION A : PRELIMINARY AND GENERAL		0
		SECTION A : PRELIMINARY AND GENERAL		0
	1200 AA & 1200 AB	SECTION A : PRELIMINARY AND GENERAL		0
A1	8.3			0
A1.1	8.3.1	Contract requirements	Sum	0
	8.3.2	Establishment of Facilities on the Site:		0
A1.2	PSAB4.1	For Engineer: as per SABS 1200 AB and PSAB	Sum	0
A1.3		Nameboard	Sum	0
A1.4		For Contractor: as per SABS 1200 AA, PS 6 and PS7.2, but	Sum	0
A1.5		Plant	Sum	0
A2	8.4	TIME-RELATED ITEMS		0
A2.1	8.4.1	Contractual Requirements	Sum	0
	8.4.1	Operation and Maintenance of Facilities on Site: for "duration		0
A2.2	PSAB4.1	For Engineer: as per SABS 1200 AB an PSAB	Sum	0
A2.3		Nameboard	Sum	0
A3	SABS 1200AA	PROVISIONAL AND PRIME COST SUM ITEMS		0
A3.1		SUMS STATED PROVISIONALLY BY THE ENGINEER		0
	8.5(a)	FOR WORK TO BE EXECUTED BY THE CONTRACTOR, a		0
A3.1.1		Survey Beacons		0
A3.1.1.1	PSAA2.4 8.8.5(b)	Temporary protective works	Sum	1
A3.2	8.6	PRIME COST ITEMS		0
A3.2.1	PSAA2.2	Special Risk Insurance as per C38	Sum	1
A3.3.	PS10	Spoil Sites		0
A3.3.1		Spoil Fees	Sum	1
A3.3.2		Overheads, charges and profit on Item A3.3.1	%	10
		SECTION A : PRELIMINARY AND GENERAL		0
		SECTION B : WATER RETICULATION		0
		SECTION B : WATER RETICULATION		0
	SABS 1200DB	EARTHWORKS (PIPE TRENCHES)		0
	8.3.1	Clear Site : See Section G		0
B1	8.3.2(a) PSDB3.1	EXCAVATION		0
		Excavate in all materials for trenches, backfill, compact and		0
B1.1		Over 50 mm dia. tot 160 mm dia. for total depth:		0
		Exceeding but not exceeding		0
B1.1.1		- 1,0 m	m	2060
B1.1.2		1,0 m 1,5 m	m	275
		Over 160 mm dia. to 200 mm dia. for total depth:		0
		Exceeding but not exceeding		0
B1.1.3		- 1,0 m	m	0
B1.1.4		1,0 m 1,5 m	m	580
B2	8.3.3	EXCAVATION ANCILLARIES		0
B2.1	8.3.3.1 (a)	Make up deficiency in backfill material:		0
B2.1.1		a) From other excavations on site (PROVISIONAL ITEM)	m <sup>3</sup>	0
B2.1.2		b) From commercial sources	m <sup>3</sup>	210
		SECTION B : WATER RETICULATION		0
		SECTION C : SEWER RETICULATION		0
	SABS 1200 DB	SECTION C : SEWER RETICULATION		0
	8.3.1	Clear Site : See Section G		0
C1	PSDB 3.2	EXCAVATION		0
		Trench excavation in all material, backfilling, compacting and		0
C1.1		For depths:		0
C1.1.1		0,00 - 1,00 m	m	0
C1.1.2		1,01 - 1,50 m	m	240

Figure 38 Extract from the Bill of Quantities



After the data was obtained it was imported into the Access database and stored in a table called 'ImportedExcel', keeping the same layout as in the Excel format. A snapshot of the table in the database is shown in Figure 39.

Item	Payment	Description	PrintControl	Unit	Quantity	Rate
		SECTION A : PRELIMINARY AND GENERAL	H		0	0
		SECTION A : PRELIMINARY AND GENERAL	H		0	0
	1200 AA & 1200	SECTION A : PRELIMINARY AND GENERAL	N		0	0
A1	B.3				0	0
A1.1	B.3.1	Contract requirements		Sum	0	0
	B.3.2	Establishment of Facilities on the Site:			0	0
A1.2	PSAB4.1	For Engineer: as per SABS 1200 AB and PSAB		Sum	0	0
A1.3		Nameboard		Sum	0	0
A1.4		For Contractor: as per SABS 1200 AA, PS 6 and PS7.2, t		Sum	0	0
A1.5		Plant		Sum	0	0
A1.6	B.3.3 PSAA 2.3	General responsibilities and other fixed-charged obligations		Sum	0	0
	PSAA 2.4				0	0
A1.7	B.3.4	Removal of Site establishment on completion		Sum	0	0
A1.8	PSAA2.5 PSAA2.5	Deal with Access to the Works and Traffic, Accommodatic		Sum	0	0
A1.9	PSAA2.7	Deal with water on the Works		Sum	0	0
A2	B.4	TIME-RELATED ITEMS			0	0
A2.1	B.4.1	Contractual Requirements		Sum	0	0
	B.4.1	Operation and Maintenance of Facilities on Site: for "durat			0	0
A2.2	PSAB4.1	For Engineer: as per SABS 1200 AB an PSAB		Sum	0	0
A2.3		Nameboard		Sum	0	0
A2.4		For Contractor: as per SABS 1200 AA, PS6 and PS7.2, t		Sum	0	0
A2.5		Plant		Sum	0	0
A2.6	B.4.3	General responsibilities and other time-related obligations		Sum	0	0
	PSAA2.3 PSAA2.3				0	0
A2.7	PSAA2.5 PSAA2.5	Deal with Access to the Works and Traffic and Attendance		Sum	0	0
A2.8	PSAA2.7	Deal with water on the Works		Sum	0	0
A3	SABS 1200AA	PROVISIONAL AND PRIME COST SUM ITEMS			0	0

Figure 39 Bill of Quantities imported into the Access database

Conversion of the data takes place once the data is available in the database. For this example this was done using the Java Application 'Table2Xml'. The XML document resulting from this is as shown in Figure 40.

```

<?xml version="1.0" encoding="UTF-8"?>
<BillOfQuantities>
  <Section>SECTION B : INTERNAL WATER RETICULATION
    <SectionNumber />
    <Reference>PSLF4.1</Reference>
    <SectionDescription>SECTION B : INTERNAL WATER RETICULATION
  </SectionDescription>
  <Item>
    <Itemno />
    <Reference>8.3.1</Reference>
    <Description>Clear Site : See Section I</Description>
    <Unit />
    <Quantity>0.0</Quantity>
    <Rate>0.0</Rate>
  </Item>
  <SubSection>EXCAVATION
    <SectionNumber>B1</SectionNumber>
    <Reference />
    <SectionDescription>EXCAVATION</SectionDescription>
    <Item>
      <Itemno />
      <Reference />
      <Description>Excavate in all materials </Description>
      <Unit />
      <Quantity>0.0</Quantity>
      <Rate>340.0</Rate>
    </Item>
  </SubSection>
  <SubSection>EXCAVATION ANCILLARIES
    <SectionNumber>B2</SectionNumber>
    <Reference />
    <SectionDescription>EXCAVATION ANCILLARIES</SectionDescription>
    <Item>
      <Itemno>B2.1.1</Itemno>
      <Reference />
      <Description>a) From other excavations on site </Description>
      <Unit />

```

```

                <Rate>0.0</Rate>
            </Item>
        .
        .
        .
        .
    </SubSection>
</Section>
</BillOfQuantities>

```

**Figure 40 Bill of Quantities converted to XML**

The next step was to confirm that the transformation process returned a XML document with a structure conforming to the structure specified by ActionIT for a Bill of Quantities. This was done by handing the XML document as well as the ActionIT XSD to the Xerces parser. The process was accomplished by executing the Java 'SchemaTest' application. After successfully parsing the document, the document is ready for report generation.

Two methods for report generation, a XSLT transformation converting the validated XML document to HTML and a Java application, were used.

The first was achieved by creating a XSL stylesheet (Figure 41) using CookTop2.200.

```

<?xml version="1.0"?>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

```

```

<xsl:output method="html" indent="yes"/>

<xsl:template match="BillOfQuantities">
  <html>
  <body>
  <br></br>
  <table border="1" cellpadding="2" width="100%" BGCOLOR="#AGDG70">
    <tr>
      <td width="2%"></td>
      <td width="100%"><center><h1 colour ="green">Section
        Summary: Bill of Quantities</h1></center></td>
      <td width="2%"></td>
    </tr>
    <tr><td width="20%"></td>
      <td width="60%">
      <td width="20%"><h3>Amount</h3></td>
    </tr>
    <xsl:apply-templates select = "Section" />
  </table>
  </body>
  </html>
</xsl:template>

<xsl:template match="Section">
  <tr><td width="20%"><h3>Section</h3></td>
    <td width="100%"><b><xsl:value-of select="child::SectionDescription"
      /></b></td>
  </tr>
  <xsl:apply-templates select = "SubSection" />

```

```
</xsl:template>

<xsl:template match="SubSection">
  <tr><td width="20%">SubSection <xsl:value-
    of select="child::SectionNumber" /></td>
    <td width="60%"><xsl:value-of select="child::SectionDescription"
      /></td>
    <td width="20%"><xsl:value-of select="sum(child::Item/Rate)" /></td>
  </tr>
</xsl:template>

</xsl:stylesheet>
```

**Figure 41 XSL Stylesheet**

This stylesheet as well as the XML document was then handed to the Xalan XSLT processor again using CookTop2.200. The processor then transformed the XML document to a HTML page, which could be displayed in a browser as shown in Figure 42.

The screenshot shows a web browser window with the following table content:

Section Summary: Bill of Quantities		Amount
<b>Section</b>	SECTION B : INTERNAL WATER RETICULATION	
SubSection B1	EXCAVATION	1742.67
SubSection B2	EXCAVATION ANCILLARIES	8045.7
SubSection B3	BEDDING (PIPES)	5129.65
SubSection B4	PIPES Supply, lay, joint, bed (Class C bedding), as specified, test and disinfect watermains for	4493.77
SubSection B6	CI SPECIALS AND FITTINGS FOR AC PIPES	74852.78
SubSection B7	VALVES AND HYDRANTS	374731.315
SubSection B8	THRUST BLOCKS	0
SubSection B9	VALVE AND HYDRANT CHAMBERS	5047.65
SubSection B10	SUNDRIES	8516.41
SubSection B11	BRF CONNECTIONS.	6804.46
SubSection B12	MARKER POSTS.	0
<b>Section</b>	SECTION C : INTERNAL SEWER RETICULATION	
SubSection C1	EXCAVATION	11133.8
SubSection C2	EXCAVATION AND INSTALLATION OF	10001.64

**Figure 42 XSLT Sample Report in HTML format**

The second method using Java was done by executing the 'ReportApp' application, resulting in a window displaying the chosen report as shown in Figure 43.

SELECT TYPE OF REPORT

Deliverables All  
 Status  
 Full  
 Summary  
 Deliverables All  
 Technical Deliverables  
 Technical Feasibility Deliverables  
 Professional Fee Deliverables

Deliverables	Unit		Dimension
Sites of size 200 sq m	site		Technical
Sites of size 250 sq m	site		Technical
Unit per Topstructure of size 35sq m	unit		Technical
Unit per Topstructure of size 45sq m	unit	4500.00/site	Technical
Subsidy in R18000 band	subsidy	R18000.00	Financial Funding
Subsidy in R12500 band	subsidy	R12500.00	Financial Funding
Geotech Slope	%	12%	Financial Funding
Geotech Soil	%	15%	Financial Funding
Title Deeds	Deeds	R300.00/deed	Legal Documentation
Access Roads	km	50000	Technical Feasibility
Bulk Sewer	MI	15	Technical Feasibility
Bulk Storm Water	MI	8	Technical Feasibility
Bulk Water Supply	MI	100	Technical Feasibility
Bulk Electricity Supply	kW	75	Technical Feasibility
Engineer Design	R		Technical Documents
Service Layout			Technical Expenditure
Geotechnical Report			Technical Documents
Geotechnical Fee			Financial Expenditure
Roads	km	R15000.00/km	NotAvailable
Stormwater	m	R800.00/m	NotAvailable
Streetlights	lights	R1200.00/light	NotAvailable
Sanitation	connections	R/connection	NotAvailable
Water supply	connections	R/connection	NotAvailable
Electricity	connections	R/connection	NotAvailable

Figure 43 Sample Report output of Java Application

## **11. Conclusions, Summary and Recommendations**

**11.1 Summary of Investigations.** The investigation into project management was done by first analysing the needs and shortcomings of some project management systems implemented by the South African Government. This revealed that the process of retrieving information from municipalities is still slow and inefficient. Technologies relevant to project management were then investigated and summarised. The Java programming language was identified as suitable for Internet applications and the Extensible Markup Language and related technologies for report generation. Java was used to implement an application for online data entry. This was accomplished by creating forms that could be completed and submitted online with the data being sent to a database. This led to the question, “what technologies should be used to manage the database as well as communications with the clients?” Servlets were identified and then used to manage communications and database processes. It was found that this works very well as the servlets provide an easy way for managing the database and for creating dynamic web pages. It is also easy to install on the server.

Difficulty experienced with the online data gathering process was with the mapping of data from the forms to existing accounts, necessitating the construction of a universal account register. The register was used to map the fields received from the forms to corresponding accounts. This however means that when a new form is being created the field names in the form must correspond to account name references in the universal account register.

Java was again used to implement an application for the aggregation of project data. This was done by using the ActionIT Enterprise data model and formatting the data accordingly. It was found that this approach worked well. Although this application works only for the ActionIT Enterprise data model, as it uses the ActionIT ‘Dimension’ attributes of the accounts in the database to construct the trees, it can easily be changed to accommodate other formats.

The Extensible Markup Language and related technologies, which included XSL, XSD, XSLT and complementing software, were researched next and also summarised. It was found that within a few steps and a bit of programming, project



data could be transformed from tables in a database to any kind of project report. The research into this field led to the discovery that for each different set of data, for example a Bill of Quantities and a Application Form, different XSL stylesheets need to be programmed to change the structure of the data to conform to a specified structure. It was found that once this was done a general stylesheet for each different kind of project report could be programmed. This stylesheet could then be used to convert the resulting XML file to a specific project report in HTML format. The only requirement for this method of report generation was found to be that the input data format should be consistent. This is required as the XSL stylesheet is programmed to search for certain fields in the data, which if changed, the stylesheet would not be able to find.

As an alternative, report generation programmed as a Java application was done. Reports were generated from data conforming to the ActionIT Enterprise data model, displaying summaries of the data in a table sorted according to the ActionIt 'Dimension' attribute of the accounts in the database.

**11.2 Conclusions.** The use of a web application for the gathering of online project data instead of a paper based system works well. The web application demonstrated the fast and efficient gathering of data as well as the management of such data. This application is viable for the gathering of project information for projects in the South African Government.

The aggregation application provided an implementation of the first version of the ActionIT Enterprise Data Model. It demonstrated the use of this model for the aggregation of projects, contributing to the validation of the model.

The demonstration of XML and its related technologies for the generation of project reports showed that these technologies could be applied advantageously for this purpose. It proved to be a simple and efficient way of generating project reports in any required format.

**11.3 Recommendations.** It was found that the creation of XSL stylesheets was made difficult by the fact that input data, for instance a Bill of Quantities, has to be examined for ways to tell if an element is a section heading, a subsection or just another element. This was needed to change the structure of a XML document to

conform to a specified structure in order for the XML document to be validated against a provided XSD schema. This process would be made much simpler, if for instance references indicating the beginning and end of specific elements are defined as shown in Figure 44. In the figure: sb = section beginning; ssb = subsection beginning; se = section end; sse = subsection end.

Reference	Item	Payment	Description	Unit	Quantity	Rate
			SECTION A : PRELIMINARY AND GENERAL			
sb		1200 AA & 1200 AB	SECTION A : PRELIMINARY AND GENERAL			
ssb	A1	8.3				
	A1.1	8.3.1	Contract requirements	Sum		
		8.3.2	Establishment of Facilities on the Site:			
	A1.2	PSAB4.1	For Engineer: as per SABS 1200 AB and PSAB	Sum		
sse	A1.3	PSAA2.7	Deal with water on the Works	Sum		
se			SECTION A : PRELIMINARY AND GENERAL			
sb			SECTION B : WATER RETICULATION			
		SABS 1200DB	EARTHWORKS (PIPE TRENCHES)			
		8.3.1	Clear Site : See Section G			
ssb	B1	8.3.2(a) PSDB3.1	EXCAVATION			
			Excavate in all materials for trenches, backfill, compact and dispose of surplus material within a freehaul distance of 0,5 km for pipes:			
	B1.1		Over 50 mm dia. tot 160 mm dia. for total depth:			
sse	B1.2	8.3.2(c)	Excavate and dispose of unsuitable material from trench bottom (PROVISIONAL ITEM)	m <sup>3</sup>		
se			SECTION B : WATER RETICULATION			

**Figure 44 Example of proposed referencing of elements**

These references can then be used to map the structure of the XML document to the desired format with less programming and in a much shorter time.

## 12. References

World Wide Web references to active web sites refer to sites as in 2001/2002.

### Articles:

1. Dare Obasanjo, An Exploration of Object Oriented Database Management Systems, 2001.
2. Rosemarie Wise, Database Types, [www.websiteowner.info/articles](http://www.websiteowner.info/articles), 25 December 2001.
3. Ian Stuart, XML Schema, an brief introduction, [www.lucas.ucs.ac.uk](http://www.lucas.ucs.ac.uk), 2002.

### Documents:

4. ActionIT Workgroup 1, A Conceptual Model of Projects in Government (Stellenbosch Version), March 2002.
5. C.V.Z. Smit, J.H. Muller, Sun Microsystems J2EE and Microsoft. NET, A technological comparison, University of Pretoria.
6. [www.local.gov.za/DCD/dcdlibrary/cmip/cmipindex.html](http://www.local.gov.za/DCD/dcdlibrary/cmip/cmipindex.html)
7. <http://www.iai-na.com/files/aecXML%20Framework%20R1.doc>.
8. <http://edocs.bea.com/wli/docs70/edi/>
9. <http://onjava.com/lpt/a/onjava/2001/03/29/tomcat.html>
10. <http://itcon.org/1999/2/>
11. [http://cig.bre.co.uk/iai\\_uk/iai/page4.html](http://cig.bre.co.uk/iai_uk/iai/page4.html)
12. [www.tmdc.org/cplusplus/info](http://www.tmdc.org/cplusplus/info)
13. [www.topxml.com/xsl/articles/xsl\\_transformations/default2.asp](http://www.topxml.com/xsl/articles/xsl_transformations/default2.asp)
14. [www.vbxml.com/xml/articles/xsd/kurt\\_schema2.asp](http://www.vbxml.com/xml/articles/xsd/kurt_schema2.asp)
15. [www.odmg.org/standard/standardoverview.htm](http://www.odmg.org/standard/standardoverview.htm)

**Web references for standards:**

16. <http://www.w3.org/TR/1999/REC-xpath-19991116>
17. <http://www.w3.org/TR/SOAP>
18. <http://www.w3.org/TR/xmlschema-0/>
19. <http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/>
20. <http://www.w3.org/TR/xmlschema-2/>

**Books:**

21. Brett McLaughlin, *Java and XML*, O'Reilly&Associates Inc., 2000
22. William Stallings, *Data and Computer Communications*, Prentice Hall, 2000
23. Daniel J. Berg, J. Steven Fritzing, *Advance Techniques for Java Developers*, Wiley Computer Publishing, 1999
24. Hiroshi Maruyama, Kent Tamura, Naohiko Uramoto, *XML and Java*, Addison Wesley Longman Inc., 1999
25. Forrest Houlette, Ph. D., *SQL A Beginner's Guide*, McGraw-Hill, 2001
26. Bruce Eckel, *Thinking in Java*, Prentice Hall, 1998
27. *Civil engineering procedure*, 5th edition

**Websites:**

28. [www.cmip.co.za](http://www.cmip.co.za)
29. [www.local.gov.za](http://www.local.gov.za)
30. [www.in-forms.co.za](http://www.in-forms.co.za)
31. [www.civil.ubc.ca](http://www.civil.ubc.ca)
32. <http://itcon.org>
33. <http://cig.bre.co.uk>
34. <http://www.oasis-open.org>
35. [www.ebxml.org](http://www.ebxml.org)

36. [www.uncec.org](http://www.uncec.org)
37. [www.unedifact.com](http://www.unedifact.com)
38. [www.aecxml.org](http://www.aecxml.org)
39. [www.sun.com](http://www.sun.com)
40. [www.jdom.org](http://www.jdom.org)
41. [www.tmdc.org](http://www.tmdc.org)
42. [www.msdm.Microsoft.com](http://www.msdm.Microsoft.com)
43. [www.visualbasic.com](http://www.visualbasic.com)
44. [www.w3c.org](http://www.w3c.org)
45. [www.whatis.com](http://www.whatis.com)
46. [www.webopedia.com](http://www.webopedia.com)
47. [www.topxml.com](http://www.topxml.com)
48. [www.java.sun.com](http://www.java.sun.com)
49. <http://jakarta.apache.org>
50. [www.acm.org](http://www.acm.org)
51. [www.xmlcooktp.com](http://www.xmlcooktp.com)
52. [www.cs.cf.ac.uk](http://www.cs.cf.ac.uk)
53. [www.cplusplus.com](http://www.cplusplus.com)
54. [www.vbxml.com](http://www.vbxml.com)
55. [www.Microsoft.com](http://www.Microsoft.com)
56. [www.gotdotnet.com](http://www.gotdotnet.com)
57. [www.onjava.com](http://www.onjava.com)
58. [www.websiteowner.info](http://www.websiteowner.info)
59. [www.odbmsfacts.com](http://www.odbmsfacts.com)
60. [www.odmg.org](http://www.odmg.org)
61. [www.simba.com](http://www.simba.com)
62. [www.actionit.org.za](http://www.actionit.org.za)

**Interviews:**

63. Dieter Kopke, CMIP, 22 November 2001, Cape Town
64. J. van Zyl, Civil Engineer, 28 July 2002, Stellenbosch

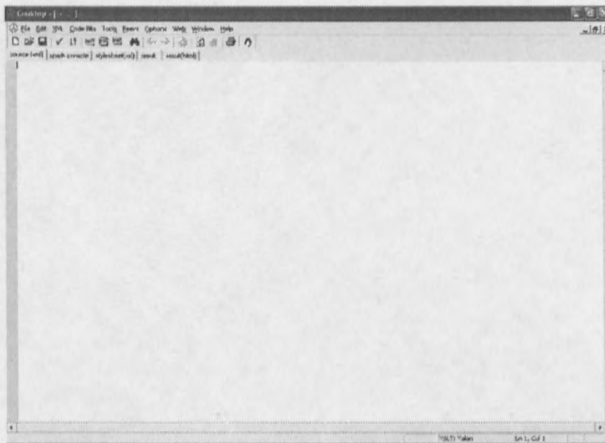
**Consulting Engineering Firms:**

- 65. Entech (Pty) Ltd.
- 66. GFJ Incorporated, Consulting Engineers and Project Managers

## Appendix A

### CookTop2.200 Description and Functionality

CookTop2.200 is a development and collaboration environment for XML/XSL and was developed by Victor Pavlov. It is distributed as freeware and can be downloaded from [www.xmlcooktop.com](http://www.xmlcooktop.com). The opening window is shown below.



CookTop provides functionality for the following [taken from CookTop's help menu]:

#### 1. XSLT processors

CookTop allows multiple XSLT processors to be used. CookTop uses an xml file to describe various processors. The file is xslt.xml and it is in the root of the configuration directory.

The selection of the XSLT processor to be used is done by Options/XSLT Engine menu. The menu lists all available processors.

## 2. Storing and retrieving XML

- **Open existing local files**

These files are opened with the help of regular file dialog. The new file can be opened in the current workspace window or in a new workspace. The user can also specify which pane is to be used for xml and xsl files.

- **Open XML via HTTP**

CookTop supports retrieving files view HTTP GET protocol. To open such a file the user must enter the URL in the URL edit box. Such a file can be edited locally but cannot be saved at the HTTP server. The user can refresh the file by pressing the refresh button. These files are saved as Windows \*.lnk files, which is the same type as the IE favourites. The user can also open \*.lnk file.

- **Workspace files**

The workspace binds together all properties of an XSLT transformation, i.e. the source, the stylesheet files, and the associated attributes like directors.

- **Workspaces auto-create**

CookTop creates automatically a new workspace every time there's new combination of source and stylesheet files. These workspace files are stored in CookTop's configuration directory: config/last. These files are also on the MRU list.

## 3. Working with XML

- **Validation**

Validates XML and XSL files. Note that this is validation for syntax correctness of the XML. It doesn't validate XPATH expressions.

- **Formatting**

Cooktop uses tidy for formatting. It indents the xml's element in order to make



it more readable. Other formatters can be plugged through the tools subsystem.

- **Encoding**

Cooktop doesn't modify the source based on encoding. The support is via the font's character set to display the encoding correctly.

- **Syntax Colouring**

Cooktop provides syntax colouring for XSL, DTD and HTML namespaces.

- **Code Bits**

Code bits are reusable code pieces organised in categories. They are easily extendable either via add new code bits dialog, or directly modifying the templates.xml. These code bits are available through context sensitive menus or through hot keys. The map of all keys is available through the Help menu. Another feature of the code bits system is that the code bits can insert code around selected xml. For example the user can select in the editor some XML and insert `xsl:for-each` element. The selected xml will end up as child of the `for-each` element.

- **Document's XML Elements**

This dialog is present in the source window and it's invoked by pressing `Ctrl+1`. It lists the unique elements of the current XML. Actually this is the document DTD derived from current document and presented in easy to use form.

- **Document's XML XPATHs**

This dialog is present in the stylesheet window and it's invoked by pressing `Ctrl+1`. It contains the Source XML unique XPATHs. Actually this is the document DTD derived from current document and presented in easy to use form.

- **Structure Navigator**

The Structure Navigator is present in the source and the stylesheet windows and it's invoked by pressing Ctrl+3. It displays the current structure of the document and let the user to jump to the selected element.

- **Named Pair Tags**

It is inline editor, invoked by Ctrl+2 , that let the user to enter a new tag with its opening and closing part. If there's selected text, it would become a child of the new element.

- **Quotes, Comments and CDATA**

Lets the user encompass text with quotes, comments or CDATA elements.

- **Open Include File**

When the cursor is positioned over an href element it opens the file referenced by href.

- **XSLT capabilities stylesheet**

Displays the capabilities of the currently selected XLST engine.

#### 4. Working with Schemas

- **DTD Editor**

#### 5. Transforming XML

- **Support for included files**
- **Run-time Parameters**
- **Transformation Raw Output**
- **Transformation HTML Output**
- **Transformation HTML Output to default external browser**
- **Transformation Output to file**
- **Support for output HTML's external references**

## **6. Editor**

- **Setting Font and Charset**
- **Find and Replace**
- **Bookmarks**
- **Changing text's case**

## **7. XPath Console**

XPath Console is environment to test Xpath against the current XML.

- **nodes**
- **value**
- **values**

## **8. Code Bits**

- **How it works**
- **XML**
- **Add new bits**
- **Assigning Hot-Keys**

## **9. Viewers**

- **XML Definition**

## **10. External Tools**

- **XML Definition**
- **Invoking external tools**

## **11. Web Browser**

- **Favourites**
- **Custom settings**
- **New window control**
- **Anchor windows**

## **12. Trace Window**

- **What is traced**
- **Working with the trace window**

## **13. Peer to Peer**

- **Peers**
- **Exchange Code Bits and Favourites**
- **Using peer's goodies**

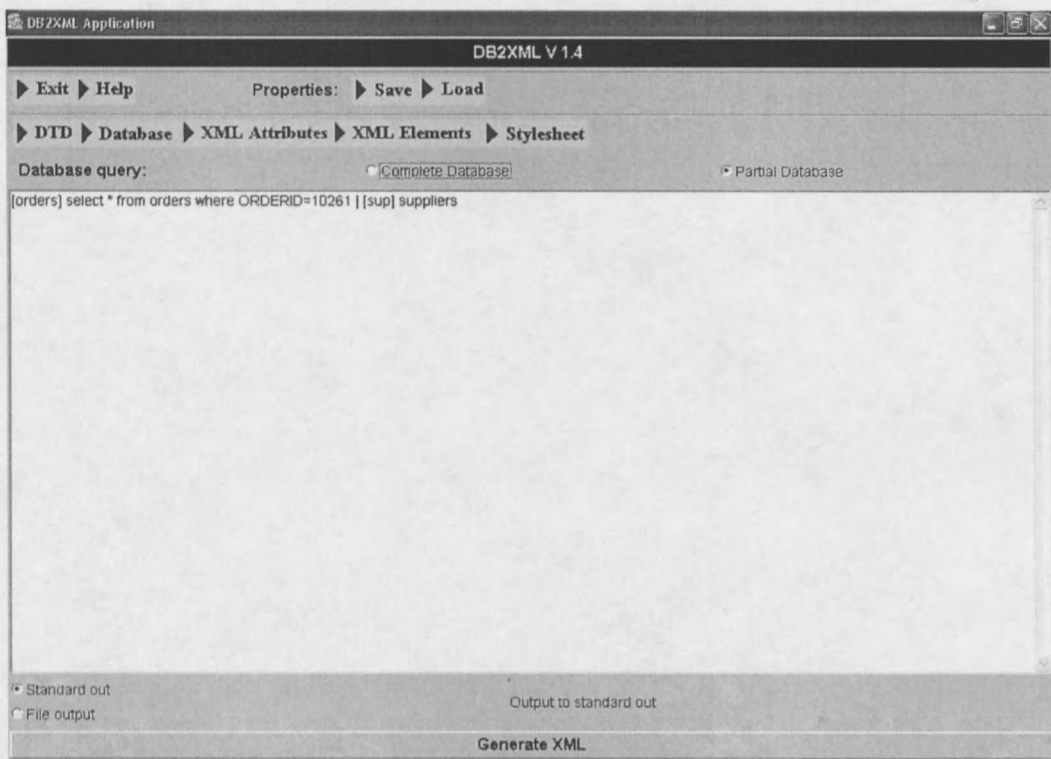
## **14. CookTop's HTTP Server**

- **Purpose**
- **Settings**

## Appendix B

### DB2XML 1.4 Description and Functionality

DB2XML is a freeware application written in Java and used for converting data from a relational database to XML format. It is available in a binary form, 'db2xmlbin.zip' and can be downloaded from [www.informatik.fh-wiesbaden.de](http://www.informatik.fh-wiesbaden.de). Included in the distribution is a XSLT processor and a XML parser. The graphical user interface is as shown below.



#### DB2XML provides functionality for:

1. Transforming the results of database queries or complete databases into XML documents or into HTML documents using XSLT stylesheets.
2. Providing attributes describing characteristics of the data (i.e. meta data).
3. Easy integration of XSLT stylesheet processors.

**and can be used:**

1. as a standalone tool (with GUI or command line),
2. as a servlet to dynamically generate XML-documents.
3. using the DB2XML API.

The generated XML is represented as a W3C DOM object or by DB2XML specific data structure. These objects can also be accessed as streams or as byte arrays.

DB2XML comes with an easy to use graphical user interface and accesses databases using JDBC drivers. It requires JDK 1.1.x and a database with a JDBC driver (or a ODBC driver using the JDBC-ODBC bridge).

Information on the installation, how to use, properties and much more is available on DB2XMLs home page,

<http://www.informatik.fh-wiesbaden.de/~turau/DB2XML/index.html> .

[Taken from DB2XML 1.4, Transforming relational databases into XML documents]

# **Appendix C**

## **Procedures for Installation of Implemented Applications and System**

The implemented system being described in this document consists of three parts, a web application combined with a server and servlets to gather and store data, a java application for aggregation of files, another java application for report generation as well as a XML application for the same purpose.

This appendix is laid out in this order, describing how to deploy the applications as well as where references assisting in the deployment of some of the system components can be found.

### **1. Web application**

The web application consists of web pages and corresponding servlets, a server and a database.

#### **1.1 Web pages**

The web pages consist of the following files:

- ApplicationForm.htm
- LevelofServiceQuestionnaire.html
- Webpage.html
- Index.html
- Frames.html
- Progress.html

These files must be copied into the Apache Tomcat 4.0/ webapps/servlet directory.

## **1.2 Server**

The server being used is the Apache Tomcat 4.0 server, which can be downloaded for free from <http://jakarta.apache.org>. Documentation and examples for installing and configuring the server, as well as documents on the deployment of web applications to the server are available at <http://onjava.com/lpt/a/onjava/2001/03/29/tomcat.html>.

The main directories are the webapps/servlet/WEB-INF/class directory where the servlets must be stored and the webapps/servlet directory where all the web pages and other files that must be retrievable by the client must be stored.

## **1.3 Servlets**

The servlets deployed to the server are:

- ApplicationFormServlet.java
- FormServlet.java
- ProgressServlet.java

These must all be copied into the Tomcat webapps/servlet/WEB-INF/class directory.

## **1.4 Database**

The database used for this system is the Microsoft Access 2000 database and is called DB2. The database must be specified as an ODBC Data source in order for the servlets to be able to connect to the database. This is done as described below:

1. In the control panel, select the ODBC Data Sources.
2. Select the System DSN tab and click 'add'.
3. Select the type of database driver and click 'finished'.
4. Supply the database name, for example DB1 and click select.



5. Browse to the directory containing the database, select it and click 'ok'.
6. Clicks 'ok' again and check that the specified database is now listed as a data source.
7. To finish the registration click 'ok'.

The database consists of the following tables:

1. **ApplicationTables** – this table must be constructed with a default table as the first entry and a column name as shown below.

Table
Default
987ApplicationData
88976ApplicationData

2. **xxxxApplicationData** – this table is constructed dynamically.
3. **Passwords** – the table needs to be constructed, containing two columns: **Username** and **password**. Values are automatically filled in by the servlets.
4. **xxxxxAccountRegister** – Must be constructed with columns: **Reference**, **Deliverable**, **Unit**, **Value per unit**, **Dimension** and **Balance**. Values are automatically filled in by the servlets.
5. **Mapping2** – This table must be constructed with two columns: **Field** and **AccountName**. Values must be filled in as shown below, mapping a Field to an accountName. The field references mapping to an account name must be assigned when an account is added to the list of universal accounts for the first time. The fields in the web pages must then be assigned accordingly.

Field	AccountName
1	Last Modified
2	Application Receive Date
3	Application Acknowledge Date
4	Project Name
5	PFD Reference number

When setting up the database these tables must be created and saved with exactly the same table names as specified above. This is needed as the servlets refer to the tables by these names.

## 2. Aggregation application

To aggregate two project data sets, the data must first be put into the ActionIT Enterprise Data Model format, as shown in the figure below, or be available as a XML document.

Reference	Deliverable	Unit	Value per unit	Dimension	Balance
f40	Access Roads	km	50000	Technical Feasibility	100
f41	Bridging Finance	R	1	Financial Funding	1000000
f42	Bulk Electricity Supply	kw	75	Technical Feasibility	2500
f43	Bulk Sewer	MI	15	Technical Feasibility	1005
f44	Bulk Storm Water	MI	8	Technical Feasibility	2100
f45	Bulk Water Supply	MI	100	Technical Feasibility	4800
f46	Electricity	connections	R/connection	Technical Installations	100

Aggregation is then accomplished by running the AggregateGui application. When the application executes, it output a XML file containing the aggregate of the two tables or XML documents and save it in the directory specified by the user.

### **3. Data Processing and Report Generation System**

#### **3.1 Java application**

This application consists of the following classes:

- DbConnector
- TownData
- ReportGui
- ReportApp
- Municipality
- TableDisplay

These must all be copied to the same directory. The database tables to be used for the reports must be specified in the DbConnector class and the 'ReportApp' must then be executed.

#### **3.2 XML application**

For this step the following is needed:

- Software for converting data from a database to XML. DB2XML and a Java application was used for this purpose.
- Parser- The Xerces2 parser was used, but any parser can be used.
- A XSLT processor, for this system the Xalan processor was used.
- XML text editor- CookTop2.200 was used.
- XSL stylesheets for changing the documents structure to the desired structure.
- XSD for validating the changed document.

The process is started by using DB2XML or the Java application, specifying all the parameters needed for DB2XML to work as described in Appendix B, or the name of the table to be converted by the Java application. The resulting XML file together with the XSL stylesheet is then passed to the XSLT processor when DB2XML is used. This can be done using CookTop2.200 as described in Appendix A. The transformed XML file resulting from DB2XML or the XML file resulting from the Java application can then be validated against the provided XSD using Xerces2 and a Java application. This is done as shown below.

```
import org.apache.xerces.parsers.DOMParser;
import java.io.File;
import org.w3c.dom.Document;

public class SchemaTest {
public static void main (String args[]) {
File docFile = new File("ActionITExample.xml");
try {
DOMParser parser = new DOMParser();
parser.setFeature("http://xml.org/sax/features/validation", true);
parser.setProperty(
"http://apache.org/xml/properties/schema/external-noNamespaceSchemaLocation",
"ActionITSchema.xsd");
parser.setFeature("http://xml.org/sax/features/namespace", false);
ErrorChecker errors = new ErrorChecker();
parser.setErrorHandler(errors);
parser.parse("docFile");
} catch (Exception e) {
System.out.print("Problem parsing the file- "+e.getMessage());
}
}
}
```

This process can also be done using CookTop2.200.

The next step is to hand the validated XML file together with a XSL stylesheet used to create a specific HTML report to the XSLT processor. Again this can be done using CookTop2.200 and the result is a HTML file showing the desired report that can be displayed in a browser.

## Appendix D

### List of Abbreviations and Acronyms

#### A

aecXML	Architecture, engineering and construction XML
API	Application Programmers Interface

#### C

CMIP	Consolidated Municipal Infrastructure Programme
COM	Component Object Model
CORBA	Common Object Request Broker Architecture

#### D

DBMS	Database Management System
DCOM	Distributed Component Object Model
DOM	Document Object Model
DTD	Data Type Definition

#### E

ebXML	Electronic business XML
UN/EDIFACT	Electronic Data Interchange For Administration, Commerce and Transport

#### I

IFC	Industry Foundation Classes
IP	Internet Protocol

#### J

J2EE	Java2 Enterprise Edition
JDBC	Java Database Connectivity
JDOM	Java Document Object Model

H

HTML      Hypertext Markup Language  
HTTP      Hypertext Transfer Protocol

M

MIME      Multipurpose Internet Mail Extensions

O

OASIS      Organisation for the Advancement of Structured Information Systems  
ODBC      Open Database Connectivity

R

RMI      Remote Method Invocation  
RPC      Remote Procedure Calls

S

SABS      South African Bureau of Standards  
SQL      Structured Query Language  
STEP      Standard for the Exchange of Product model data

T

TCP/IP      Transmission Control Protocol/Internet Protocol

W

W3C      World Wide Web Consortium

X

XML      Extensible Markup Language  
XSD      XML Schema Definition  
XSL      Extensible Stylesheet Language  
XSLT      Extensible Stylesheet Language Transformations



## **Appendix E**

### **Reference to Data CD Content**

A CD is provided with all the software necessary to demonstrate the applications and system implemented in this thesis. The CD contains three folders:

- Web application
- Aggregation
- XML report generation

Readme.txt files explaining the layout of the files and how it should be implemented are provided.