Towards a unified methodology for supporting the integration of data sources for use in web applications

A thesis submitted for the degree of
Doctorate of Information Technology

By
Jeremy Nunn

Murdoch University
2016
I declare that this thesis is my account of my research and contains as its main content work which has not previously been submitted for a degree at any tertiary education institution.

Jeremy Nunn
Abstract

Organisations are making increasing use of web applications and web-based systems as an integral part of providing services. Examples include personalised dynamic user content on a website, social media plug-ins or web-based mapping tools. For these types of applications to have maximum use for the user where the applications are fully functional, they require the integration of data from multiple sources. The focus of this thesis is in improving this integration process with a focus on web applications with multiple sources of data.

Integration of data from multiple sources is problematic for many reasons. Current integration methods tend to be domain specific and application specific. They are often complex, have compatibility issues with different technologies, lack maturity, are difficult to re-use, and do not accommodate new and emerging models and integration technologies. Technologies to achieve integration, such as brokers and translators do exist, but they cannot be used as a generic solution for developing web-applications achieving the integration outcomes required for successful web application development due to their domain specificity. It is because of these difficulties with integration, and the wide variety of integration approaches that there is a need to provide assistance to the developer in selecting the integration approach most appropriate to their needs.

This thesis proposes GIWeb, a unified top-down data integration methodology instantiated with a framework that will aid developers in their integration process. It will act as a conceptual structure to support the chosen technical approach. The framework will assist in the integration of data sources to support web application builders. The thesis presents the rationale for the need for the framework based on an examination of the range of applications, associated data sources and the range of potential solutions. The framework is evaluated using four case studies.
# Table of contents

Abstract......................................................................................................................... 3
Table of contents........................................................................................................... 4
List of tables.................................................................................................................. 10
List of figures................................................................................................................ 12
Definitions...................................................................................................................... 16
UML figures legend....................................................................................................... 16
Acknowledgements....................................................................................................... 17

1 Introduction .............................................................................................................. 18
  1.1 Overview .............................................................................................................. 18
  1.2 Integration of data sources for web applications .................................................. 19
  1.3 Support for developing web applications .............................................................. 21
  1.4 Research problem ............................................................................................... 22
  1.5 Aims ...................................................................................................................... 25
  1.6 Research questions ............................................................................................. 26
  1.7 Structure of this thesis ....................................................................................... 28
2 Literature Review ..................................................................................................... 30
  2.1 Background .......................................................................................................... 30
  2.2 Web applications and data sources ...................................................................... 31
  2.3 Integrating data sources for use in web-based applications .................................... 35
    2.3.1 Definition ...................................................................................................... 35
    2.3.2 Web applications types ............................................................................... 37
    2.3.2.1 Web application feeds .......................................................................... 38
    2.3.2.2 Application programming interface ....................................................... 38
    2.3.2.3 Web apps .............................................................................................. 39
    2.3.2.4 Summary .............................................................................................. 40
    2.3.3 Classification ............................................................................................... 40
    2.3.3.1 Data translation dimension ................................................................... 43
    2.3.3.2 Programmable dimension .................................................................... 45
    2.3.3.3 Centralised dimension .......................................................................... 46
    2.3.3.4 Meta-data dimension ............................................................................. 47
4.2.4 Instrument

4.2.5 Administration of the methodology

4.3 Framework

4.3.1 Overview

4.3.2 Structure

4.3.3 Applying the framework to any integration scenario

4.3.4 Framework supporting requirements

4.4 Summary

5 Applying GIWeb to case studies

5.1 Overview

5.2 Application to the Zend Framework

5.2.1 Overview

5.2.2 Case study

5.2.3 Problems and technology options

5.3 Application of GIWeb

5.3.1 Conceptual overview

5.3.2 Implementing the GIWeb plug-in

5.3.3 Implementation

5.3.4 Implications

5.4 Application to the PRINCE2 framework

5.4.1 Overview

5.4.2 Scenario

5.4.3 Problems and technology options

5.5 Application of GIWeb

5.5.1 Conceptual overview

5.5.2 Implementing the GIWeb plug-in

5.5.3 Implementation

5.5.4 Implications

5.6 Application to the Eclipse platform

5.6.1 Overview

5.6.2 Scenario

5.6.3 Problems and technology options

5.6.4 Application of GIWeb

5.6.5 Conceptual overview
7.3 Major contributions .............................................................................. 264
7.4 Future work ........................................................................................ 269
7.5 Closing comments ............................................................................. 270
Appendix 1 ............................................................................................. 271
Appendix 2 ............................................................................................. 272
Appendix 3 ............................................................................................. 277
References ............................................................................................. 280
List of tables

Table 2.1 Classification Scheme Dimensions for Data Integration .......................... 50
Table 2.2 Common data model work flow dimensions .............................................. 55
Table 2.3 Compares this model with the dimensions of the classification scheme ... 58
Table 2.4 Language model dimensions .......................................................... 60
Table 2.5 Schema mappings dimensions .................................................... 64
Table 2.6 Tagging approach ........................................................................ 66
Table 2.7 Categorisation of data stream elements ............................................. 66
Table 2.8 Data stream meta-tagging model dimensions .................................... 67
Table 2.9 Layer models dimensions .......................................................... 71
Table 2.10 Model comparison against classification scheme ............................... 73
Table 2.11 Standardisation of data ............................................................... 79
Table 2.12 Standardisation dimensions comparison ........................................... 80
Table 2.13 Broker characteristics by 5 layers .................................................. 83
Table 2.14 Broker dimensions comparison ................................................... 85
Table 2.15 Extensible languages dimensions comparison ................................ 88
Table 2.16: A sample class ........................................................................ 89
Table 2.17 Exposed APIs dimensions comparison ........................................... 91
Table 2.18 Public APIs dimensions comparison ............................................. 93
Table 2.19 Extract transform load dimensions comparison ................................. 95
Table 2.20 Application comparison against classification scheme ....................... 98
Table 2.21 Extending existing frameworks ...................................................... 102
Table 2.22 Adapting an IT management framework ........................................ 104
Table 2.23 Adapting a software development framework .................................. 108
Table 2.24 Altering toolkits ...................................................................... 111
List of figures

Figure 2.1 Three tiered client / server ................................................................. 33
Figure 2.2 UML diagram of web application architecture ................................. 34
Figure 2.3: UML component diagram of the data translation process ................. 44
Figure 2.4: UML component diagram of programmable data work flow ........... 46
Figure 2.5: Common data model work flow ......................................................... 53
Figure 2.6 Attribute mapping model example ..................................................... 56
Figure 2.7 An example data mapping scenario .................................................... 57
Figure 2.8 Schema breakdown model ................................................................. 62
Figure 2.9 Tagging model work flow ................................................................. 65
Figure 2.10 Layer model work flow for data integration ...................................... 70
Figure 2.11 UML component diagram of broker work flow ............................... 82
Figure 2.12 The workflow between brokers and web services ............................ 84
Figure 2.13 Semantic web elements ................................................................. 86
Figure 2.14 UML Composite structure diagram of exposed API work flow ......... 90
Figure 2.15 Public APIs workflow ................................................................. 92
Figure 2.16 UML component diagram of extract transform loads work flow ....... 94
Figure 2.17 New framework from extensions .................................................... 101
Figure 2.18 Adapting a framework to an integration activity .............................. 103
Figure 2.19 Extending existing software development frameworks .................... 107
Figure 2.20 Illustrates an example extending the DataGrid toolkit .................... 110
Figure 3.1 UML component diagram integration technology decision in the web development process .............................................................. 123
Figure 3.2: UML activity diagram workflow problems upon choosing an Integration technology ........................................................... 124
Figure 3.3: UML activity diagram of the current workflow of a data integrator ... 126
Figure 3.4: UML activity diagram showing support framework that can be applied to different integration scenarios and facilitate different technologies………………. 127

Figure 3.5: UML activity diagram of methodology instantiated through a framework………………………………………………………………………………… 130

Figure 3.6 UML component diagram of requirements identification in the web development process………………………………………………………………………………… 131

Figure 4.1: UML component diagram of high-level overview of the methodology and framework workflow …………………………………………………………………… 147

Figure 4.2 UML component diagram of the pre-initiation activities …………… 149

Figure 4.3: UML component diagram of the methodology for supporting the integration of data sources for web applications………………………………………. 150

Figure 4.4 Sequence of phases of the methodology for supporting the integration of data sources for web applications………………………………………………………………… 151

Figure 4.5: UML component diagram analysis phase of workflow activities to classification……………………………………………………………………………….. 152

Figure 4.6: UML component diagram of the development phase activities……… 155

Figure 4.7: UML component diagram of test phase activities…………………… 156

Figure 4.8: UML component diagram of the quality phase activities……………. 158

Figure 4.9: Tree diagram of support framework……………………………………. 162

Figure 4.10: Tree diagram of available technologies……………………………… 164

Figure 4.11 API characteristics workflow ……………………………………………. 166

Figure 4.12 Technology dimensions tree workflow ………………………………… 167

Figure 5.1 UML Component diagram of retail web application…………………. 177

Figure 5.2 Zend Framework in the API workflow ………………………………… 180

Figure 5.3 APIglility dashboard ………………………………………………………… 181

Figure 5.4 Create new API ………………………………………………………………. 182

Figure 5.5 Choosing between Rest services of RPC services……………………. 183

Figure 5.6 Define the web application…………………………………………………. 184

Figure 5.7 shows defining the data sources using the wizard…………………… 185
Figure 6.5 Bar Graph comparison of PRINCE2 model number of MP outputs and work packages ................................................................. 236

Figure 6.6 Bar and line graph comparison of number of attempts versus potential errors ............................................................ 246

Figure 6.7 UML Activity diagram current approaches ........................................... 255

Figure 6.8 UML Activity diagram maturing approaches ...................................... 256
Definitions

The following definitions are used to give context to common used terms in this thesis.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approach</td>
<td>In this work, an approach is a technical solution or method</td>
</tr>
<tr>
<td>Data</td>
<td>Information in raw or unorganised form</td>
</tr>
<tr>
<td>Data Integrator</td>
<td>In this work, the term ‘data integrator’ is used to describe the role of a web developer, blog builder, or middleware programmer in bringing together various forms of data to make a smoothly functioning application</td>
</tr>
<tr>
<td>Framework</td>
<td>A conceptual structure describing a system, concept, information flow or data flow</td>
</tr>
<tr>
<td>Method</td>
<td>In this work, a method is a systematic process for accomplishing or approaching a task</td>
</tr>
<tr>
<td>Methodology</td>
<td>In this work, methodology is a set or system of methods, principles, and rules for regulating the integration of data sources for use in web applications</td>
</tr>
<tr>
<td>Model</td>
<td>A representation of information flow defining how data and information is connected to each other and how it is processed and stored</td>
</tr>
<tr>
<td>State of art</td>
<td>Current state of the industry field and domain</td>
</tr>
<tr>
<td>Web App</td>
<td>Web Application</td>
</tr>
<tr>
<td>Web</td>
<td>World Wide Web</td>
</tr>
</tbody>
</table>

UML figures legend

- Activity
- Control Flow
- Object
- Decision
- Initial
- Final Node
- Required Interface
Acknowledgements

I would like to thank the many people who have provided support, guidance and encouragement during this research project, especially my supervisors, Doctor Kevin Lee, Doctor Graham Mann and Danny Toohey. I have benefited greatly from their experience and wisdom and they have helped guide my thesis journey in such a positive way.

Kevin Lee provided substantial support in helping me refine my topic and structure my research, his encouragement and confidence in my ability have been invaluable. Danny Toohey helped me grapple with thesis formatting and Graham Mann was most accommodating in taking over from Kevin Lee when he left to return home overseas. His input, experience and guidance has been most rewarding.

Finally I would like to thank my wife, Camilla Nunn who provided much encouragement and moral support.
1 Introduction

1.1 Overview

Organisations and consumers are making increasing use of web applications as an integral part of conducting business, providing or using a service over the Internet (Offutt, 2002; Nečaský, 2007; Halevy, Rajaramn & Ordille, 2006). A web application is a program that is stored on a remote server and accessed over the Internet through a web-browser (Rouse, 2011). The web is a highly programmable environment that enables the creation and deployment of web applications to allow users to retrieve data and interact with content (Acunetix, 2013). This includes the searching of data through an online database or interacting with dynamic content in a website by navigating its structure.

Web applications rely on data from heterogeneous sources also located on the web. These sources consist of different structures and formats (Cardoso & Sheth, 2006). Organisations, web developers and consumers require real-time integration, that is, the ability to combine data from different sources (Mozafari, Zeng & Zaniolo, 2010), and provide the user with a uniform view of these data even as they are being altered (Lenzerini, 2008). Such web applications include personalised dynamic user defined content on a website, social media plug-ins or web based mapping software such as Google Maps. These applications all rely on the capacity to integrate data from a variety sources to provide the user with a consistent, understandable and usable view of the data (Lenzerini, 2008). This can be problematic because the data from these sources tends to have different structures and formats (Cardoso & Sheth, 2006), and are required to be integrated in real-time (Mozafari et al., 2010). Furthermore, many web applications rely heavily on integration with real-time data for example, for displaying the trend between what occurred yesterday and today (Nehme & Bertino, 2009).

Solutions that have been suggested to deal with this include Extensible Markup Language (XML), Web Services, Data Warehousing and Service-Oriented Architectures (SOA) (Geroimenko, 2003; Wolk, 2006). These solve specific integration problems by extracting data, transforming it and offering the data a single consistent view (Wolk,
They are domain-specific solutions (Wolk, 2006) rather than appealing to broader integration goals. Other approaches, such as brokers and translations achieve integration, but only in particular scenarios (Wolk, 2006; Songtao, 2010; Nehme, 2009). These approaches can suffer quality problems with semi-structured data, structural constraints with schemas, compatibility issues, reliance on a technical expert to implement and a high cost to maintain, lack of maturity and reusability and inconsistent reliability (McCarthy & Shawn, 1998; Gujarathi, 2011; Fletcher, 2007; Alexe, 2008; Wyss & Robertson, 2005; Rahm & Hai Do, 2000; Kahn, 2006; Lacey, 2016). Current approaches are not necessarily appropriate in all integration scenarios because of their domain and application specificity (Wolk, 2006; Guo, 2010; Nehme & Bertino, 2009).

This thesis examines solutions for the integration of data sources for use in web applications. It aims to improve web application development by proposing a methodology and supporting framework that can be applied to a wide range of integration scenarios. This methodology and framework aims to improve the process of integrating data sources for the purposes of presenting the data in a web application and providing a means of accessing the data from the various sources in a consistent, timely and accurate manner.

The remainder of this chapter is structured as follows. Section 1.2 describes the process of data integration for web applications. Section 1.3 discusses support for developing web applications. Section 1.4 defines the research problem. Section 1.5 discusses the aims. Section 1.6 focuses on the importance of this research. Finally, Section 1.7 gives an overview of the structure of this thesis.

1.2 Integration of data sources for web applications

Web applications range from custom applications, intranet software, open source and community applications which are all accessible over the web (Carr 2002). These different types of web applications share similar structural characteristics and can use data sources classified as static, dynamic, live, archived and steaming (Mozafari et al., 2010). These sources of data might consist of raw text, different media types (images, video and sound), log entries or non-meaningful data. Web applications need to access a broad range of data types ranging from zipped files, software files, raw text, meta-data,
documents, images, videos, music sourced from static areas, dynamic or streaming traffic.

Many web applications rely heavily on integration with real time data (Nehme, 2009). Examples of real time data often used in web applications include live feeds such as financial market data, the weather, status updates on Facebook or Twitter, news feeds and mapping. A method to search in real time is a web interface that finds and presents results to the user in a meaningful way. Examples of these methods include Search Engines, Web Portals, Directories, Local Networks, Public Networks, Peer to Peer Networks and Private Networks (Nations, 2012; Carr, 2002; Flannery, 2000; Tajudeen, 2012).

Data integration is a requirement for many contemporary web applications (Rajeswari & Varughese, 2009). Data Integration is important for web applications because in their very nature they rely on searching, querying or retrieving information from different data sources (Offutt, 2002). They rely on external plug-ins such as news feeds, analytic sources and statistical providers that are reliant on pooling multiple data sources together (Rajeswari & Varughese, 2009; Offutt, 2002). The primary mode for data integration at present is via web services for access, retrieval and interaction with data sources via web applications (Offutt, 2002).

Integration of different sources that a web application needs is a complex task (Guo, 2010). Troublesome factors include circumstances where different sources may refer to the same real-world entity in different ways, such as what might happen when two customer relationship management data sources in a web-based environment refer to the same customer information but the data are presented and structured very differently. The same customer could be represented completely differently in the two systems. It can be quite difficult to recognise and merge various data references that refer to the same data entity (Guo, 2010) due to redundant data from the multiple representations of the same data.

There are different ways of describing the structure of a data source such as through schema definitions or web services. Data sources may provide data generally in any representation described in the schema. Data sources and services are usually
independent of each other resulting in their own structures or schemas (Nations, 2012; Carr, 2002; Flannery, 2000; Tajudeen, 2012). Mashups integrate a set of web-services and data sources, often referred to as mashlets (Deutch, 2010).

The need for integration is further illustrated with the following examples. In a financial market, a vast array of information is collected regarding stock movements ranging from the current price, trends and predictions. This information comes from a single data source such as the Australian Stock Exchange but is spread across a string of networks accessible by many different websites, feeds, portals and devices, all presenting it in a unique format and filtering or displaying the information as required.

1.3 Support for developing web applications

The integration approaches described in this thesis to assess their viability can be divided into three groups, model approaches, technical approaches and management process-based approaches. This thesis describes these integration approaches further and assesses their viability.

Model approaches do this by clearly modeling the data processes and flows required to achieve data integration. They are a conceptual plan and are abstract in nature referring to the overall data flow structure or type (Kay, 2003). Models are used either as a source to aid data integration or as an implementation tool. Models for data integration can be structural or semi-structural and can exhibit many complex data relationships and can be issued by many formats in favor of data exchange (Jinan & Guojing, 2010). Models include the Open System Interconnection (OSI) model, Schema Models, Web Services Models, Common Information Model and Mapping Model (McCarthy & Shawn, 1998), (Kazzaz, 1997; Wyss & Robertson, 2005; Marnette & Papotti, 2010; Kühne, 2005; OSI, 2010).

Model approaches are often implemented using web application technologies. A model approach lays down the framework for how a technical approach is implemented. Technical approaches such as Standardisations, Brokers, Extensible Languages, Exposed APIs, Public APIs and Extract Transform Loads all boast strong support for data integration. The types of integration environments a web developer works in include
programming environments such as development using the Eclipse platform and Zend framework.

Management process-based approaches can be used to support the integration process. The processes consist of extending existing frameworks, adapting an IT management framework or altering software development frameworks and toolkits. Project management frameworks such as PRINCE2 help in modelling the conceptual management of an integration project.

These approaches are not always appropriate in all integrating situations. This thesis examines model, technical and management process-based approaches further and their viability as an integration approach for all integration scenarios.

1.4 Research problem

This research investigates approaches for supporting data integration for use in web applications. It analyses the range of web applications and their associated data sources along with the problems faced when integrating them. The aim is to improve web application development by defining a methodology supporting the integration of data sources for web applications such that the problems are minimised.

There are many existing models, methodologies, applications and languages for achieving integration of data sources for use in web applications. Each features a range of distinguishing characteristics that make it effective in achieving data integration as discussed in the body of literature. However, there are common deficiencies with these models as they are intended to support integration within a specific domain. They also face a series of shortfalls such as lack of maturity and reusability, compatibility issues, reliance on a technical expert and high cost to maintain.

The following examples illustrate some of these deficiencies. Mapping models face challenges when transforming and integrating meta-data between schemas resulting in compatibility issues or obstacles in achieving schema mapping (Marnette & Papotti, 2010). Another challenge is in the description of schema elements leads to interpretation
shortcomings (Guilian, 2006). Language models are often domain specific designed to support a specific purpose (Wyss & Robertson, 2005). The Common Information Model has been categorized as both costly to develop and maintain as well as developed too quickly by the vendors who create them (McCarthy & Shawn, 1998). It is heavily reliant on high-level technical implementation and technical maintenance meaning a non-technical person would struggle with these tasks. In this work, the term ‘data integrator’ is used to describe the role of a web developer, blog builder, or middleware programmer in bringing together various forms of data to make a smoothly functioning application. A non-technical data integrator would struggle to maintain and in some instances implement the models and technical approaches discussed without heavy IT involvement or a high-level technical skillset themselves.

There are common causes of problems in data integration projects such as demanding too much information, no effective versioning strategy, and no support for system-level extensions (Wolk, 2006). One of the key problems is the development in time (Klimek & Nečaský, 2010). The data structure and operations on data are changing. In a complex environment, the problem is more difficult as the applications use many data sources (Klimek & Nečaský, 2010). This can lead to challenges with the handling of data as it flows from one source to another. Many areas are able to handle this within their domain specific purposes but when requirements are outside the scope of the domain, consistency issues arise.

There is an immense amount of work and technical knowledge required to integrate with different data sources. The current systems and technologies that require real time integration such as expert systems, disaster management systems, intelligent systems, report generation and network traffic monitoring focus on why it is challenging to integrate with these specific mediums. This is done by illustrating the technical details of combining all sorts of structures with unstructured data sources to achieve integration. Technologies to achieve integration, such as brokers and translations (Wolk, 2006; Guo, 2010; Nehme, 2009), are not used as a generic solution for developing web-applications. A problem with Extract Transform Load (ETL) tools as an example is that they were designed to solve a niche problem of warehouse loading. They are not useful for general data transformation (Hellerstein, Stonebraker & Caccia, 1999) and ETL’s cannot support real time transfers (Kimball, 2008).
Common data source challenges are consistent amongst the different groups of technical approaches. These include frequency changes with data sources, load changes, dealing with unstructured databases, new technologies, format changes, trusting the reliability of integration with data, new ways to take advantage of integration using web services, indexing and combing data, queries and the semantic web (Fujun Zhu & Turner 2004; Wolk et al., 2006). Data sources are unknown at design time and may evolve (Zhu, 2004). The evolution of data sources can range from load changes, dealing with unstructured databases, new technologies, format changes, improved ways to take advantage of integration using web services, indexing and combing data, queries and the semantic web. There are versioning strategies in data integration enabling technologies (Wolk, 2006) that can cause for failure in the data integration process. A lack of a versioning strategy and that no matter how much time and effort is put into defining an XML schema as an example, will need to change over time.

Web services that require too much information are an area where challenges can arise (Wolk, 2006). Problem areas occur with how web developers develop a web service for point to point integration. Research suggests the web developer tends to think of the data their particular service needs and not the potential growth implications of that data especially as it changes and expands (Wolk, 2006). Using a web service and a schema, there is a tendency to treat all data as required. As soon as the service determines that some required data is not present, it rejects a request and the client gets a service fault (Wolk, 2006). This approach to defining web services is too rigid and leads to systems that are very tightly coupled. Any change in the service’s requirements forces a change in the clients (Wolk, 2006). These two examples highlight how schemas and web services are not as forgiving when data is not present in a client request and degrades the ability to achieve data integration.

The very nature of data lays the foundations for challenging obstacles in the data integration process (Mozafari et al., 2010). Data is stored in a variety of different types and formats such as static and real time data sources (Mozafari et al., 2010). Integrating using these diverse formats can pose a difficult challenge especially if attempting to query data in a real time, live format such as through streaming. Providing an integrated access to multiple heterogeneous sources is a challenging issue in global information
systems for cooperation and interoperability (Bergamaschi, 1999). That two fundamental problems arose, how to determine if the sources contain semantically related information, that is, information related to the same or similar real-world concepts and secondly, how to handle semantic heterogeneity to support integration and uniform query interfaces (Bergamaschi, 1999). Questions such as how fast data can be accessed, queried and presented back to the user must be asked by the researcher, developer or user integrating with the data.

Existing approaches are not always suitable for being used as an overarching framework that can be applied in all integrating scenarios. The viability of applying existing approaches to achieving integration on a wide high-level scale is quite low.

1.5 Aims

This thesis aims to improve the methods of data integration in web application development. It evaluates the viability of existing approaches. It proposes a unified methodology instantiated with a support framework. This approach will be evaluated with four case studies by developing some example applications as cases and applying the methodology proposed in this thesis to them. The research in this thesis is focused on the below aims:

1: The challenges for web application development associated with a lack of generic supporting architectures that will apply to a range of integration scenarios.

2: Helping a data integrator make decisions on how which integration approach and technologies to use for their integration activity based on their web application classification.

3: The need for a unified top-down data integration framework that can be applied to many different data integration implementations for use in web applications. The proposed methodology and its support framework will act as a guide. It’s a goal centric one with a top-down design involving working out the goal, identifying where the web application fits, applying a methodology and importing the technology to apply it.
1.6 Research questions

The questions addressed in this thesis are:

1: What challenges are currently faced when attempting integration of data sources for use in web applications?

This research question will address the first aim discussed in Section 1.5.

2: What characteristics of web applications determine what integration method to use for integrating of data sources?

This research question will address the second aim discussed in Section 1.5.

3: Can a high-level methodology be instantiated using a framework to support integration of data sources for web applications?

This research question will address the third aim discussed in Section 1.5.

The research discussed in this thesis will examine the areas posed by these three questions. It will investigate the challenges with existing approaches and determine the characteristics of web applications for determining what integration method to use when integrating data sources. It will examine through cases the applicability of a high-level methodology instantiated using a framework to support integration of data sources for web applications.

1.8 Importance of this research

The main contribution of this thesis is the identification and validation of requirements for the integration process. Accompanying this is a list of requirements that can be used to evaluate the support a data integration framework provides for successful integration with data sources for web applications.
An important theoretical outcome will be the categorisation of different applications. The categorisation of applications starts by identifying the challenges faced when integrating with data sources with each application. This is a valuable contribution to the field since it allows a developer, research or data manager to identify where their system or application fits in relation to potential integration problems and they can then apply the correct approach in successful integration with intended or potential data sources.

To demonstrate that the approach can be applied to a wide range of web applications, it will be tested for a series of web applications. Each application will be categorised based on the different criteria of challenges faced when integrating the data used in the applications when examining the body of literature in the state of art of data integration. This classification will be used to demonstrate the viability of the integration methodology against a group of common challenges faced by any number of web applications that could be categorised under that specific classification of characteristics. The aim is to enhance the data integration process by knowing how the specific web application is to be integrated with based on its characteristics and therefore apply the correct integration approach.

These contributions are valuable, since they allow a developer, researcher or data manager to identify where their system or application fits in relation to potential integration problems. They can then apply the correct approach for successful integration with intended or potential data sources. Commercially the outcome of this research could be invaluable to the industry as corporations, IT professionals and researchers must integrate data sources through so many ways and when dealing with complex data sets, the methodology, and case studies discussed in this thesis could benefit them.

In summary, the literature suggests that there is a problem with contemporary methods of data integration which implies the need for a supporting unified methodology with a support framework to solve these challenges. Types of data are growing vast and complex, making the task of anyone attempting to integrate live data sources more difficult.
1.7 Structure of this thesis

This thesis is presented in 8 chapters, as follows.

Chapter 2 introduces the field of data integration. An analysis is presented on the background of data integration, an analysis of models and technical approaches for data integration and characteristics of each. A classification scheme of model types and technical approaches is proposed. The exercise highlights the difficulties associated with data integration of existing management process-based, model and technical approaches.

Chapter 3 presents the need for a solution to aid the data integrator in their integration process based on the analysis discussed in Chapter 2. This chapter presents the need for requirements for a solution. It discusses how to identify requirements, approaches and validation of requirements. It concludes with a list of requirements.

Chapter 4 presents the proposed approach GIWeb, a methodology for supporting data integration of web applications and its support framework. The chapter describes the methodology used to achieve the objectives of this thesis instantiated through a framework. It proposes GIWeb as a methodology for data integration for web applications and a framework to support a data integrator in their integration process.

Chapter 5 focuses on the application of GIWeb in Chapter 4 to several cases. It covers different types of integration environments including programming environments such as development using the Eclipse platform and Zend framework and examines the application of GIWeb to a project management framework PRINCE2 where the conceptual management of an integration project is modeled out. The focus on varying types of integration environments serves to demonstrate how the methodology and framework interacts with the whole development process and the benefits of it. Chapter 5 also demonstrates the proposed approach through building a web application using the Zend framework. This chapter presents a scenario where an application is developed and needs to receive data and integrate it with an external server. It applies the application of GIWeb as a solution and discusses how the web developer will implement and realize this.
Chapter 6 presents an evaluation of the methodology and supporting framework. It discusses how the proposed approach supports the requirements discussed in Chapter 3. It evaluates the approach against the Integrator and System requirements. It evaluates the approach against the classification scheme proposed in Chapter 2 to demonstrate the applicability of the approach for supporting the integration of data sources for use in web applications. It evaluates the qualitative outcomes and provides an analysis of the quantitative outcomes. Several case studies are discussed to demonstrate the application of the methodology and the framework.

Chapter 7 presents an overview and conclusions of the thesis. It highlights the major contributions and discusses potential future research.
2 Literature Review

2.1 Background

This chapter presents a review of the current state-of-the-art in web development and integrating data sources for use in web applications. It provides a background to the areas of data sources, data integration and web applications. It presents an investigation into the approaches for integrating data sources for use in web applications. To address the research questions stated in Chapter 1, it is important to review the “state-of-the-art” in several areas.

This chapter proposes a classification scheme (see Section 2.3.3) that contrasts the characteristics of the commonly used approaches to data integration for web applications. The approaches to data integration for web applications can be categorised into three groups including model based approaches, technical approaches and management process-based approaches.

The classification scheme is used to analyse approaches for presenting data in web applications. It considers the viability of model, technical and management process-based approaches for the integration of data sources for use in web applications in all integration scenarios.

This chapter presents an analysis of the state-of-the-art in the viability of a supporting top-down framework for data integration to consult and aid in attempts to integrate with data sources through web applications. The remainder of this chapter is structured as follows. Section 2.2 defines web applications and data sources. Section 2.3 provides a background to integrating data sources for web-based applications. Section 2.4 presents an analysis of different architectures and models for achieving data integration such as layer models, common, schema, data mapping and language models. Section 2.5 presents an analysis of technical approaches to data integration such as standardisations, brokers, extensible languages, exposed APIs, public APIs and extract
transform loads. Section 2.6 reviews the different management process-based approaches discussing various software development processes, methodologies and frameworks such as extending existing frameworks, adapting a project or IT management framework or altering software development frameworks and toolkits. Finally, Section 2.7 draws conclusions based on the themes discussed in this chapter.

2.2 Web applications and data sources

Web applications commonly use data from both static and real time data sources. Static data sources are those that do not change, they are a snapshot of information at a given point of time (Mozafari, 2010). Real time data sources are those that are dynamic in nature in that the data is constantly changing (Mozafari, 2010).

Web applications are software based applications built to run in a web browser (Carr, 2002; Nations, 2012) connected to web-enabled and web-based applications on remote servers. They are accessed with a web browser (Carr, 2002). The web is a highly programmable environment that enables many opportunities for creation and deployment of applications to allow users to retrieve data and interact with content over the web (Acunetix, 2013). Web applications are built from the ground up to run over the web. Web-enabled applications on the other hand involve adding a web interface to software applications that were built to run in a non-web-based environment such as a computer desktop based program (Carr, 2002; Nations, 2012). Web applications use a web server to process incoming and outgoing requests from a user accessing static or dynamic pages that may or may not use a database.

A web application can be as simple as a message board or a live news feed on a website, or as complex as Google apps, Microsoft Office Live, and WebEx WebOffice (Nations, 2012). Some web applications may have been created by developers for an organisation as a means of broadening the organisations reach to their customers by enabling a web-based gateway to their other systems (Carr, 2002). Others are created by developers to be shared through a common marketplace of web applications such as Google apps (Krill, 2010).
Web applications can be easily adapted by an organisation as well as further developed through the means of application programming interfaces (API) (Krill, 2010; Orenstein, 2000). APIs allow developers an interface with the web application to access and execute it on their platform (Orenstein, 2000). An online website creation and editing tool known as Word Press, for example, features many weather web applications that act as plug-ins that retrieve the weather statistics from multiple data sources. They rely on an API to pull different weather statistics. Web APIs can be broken into two general categories: Remote Procedure Call (RPC) and REpresentational State Transfer (REST) according to Apigility (2014). RPC is characterised as a single Uniform Resource Locator (URL) which is an address to a resource on the Internet on which many operations can be called via POST (Apigility, 2014). Examples include XML-RPC and SOAP. A POST request is a method supported by the HTTP protocol used by the World Wide Web (The Internet Engineering Task Force, 2016). REpresentational State Transfer (REST) is an architecture designed around the HTTP specification (Apigility, 2014). It is the architectural style underlying the Internet (Jakl, 2005). It enables the caching and reuse of interactions, dynamic substitutability of components, and processing of actions by intermediaries, thereby meeting the needs of an Internet-scale distributed hypermedia system (Fielding, Taylor 2002). Its characteristics include the ability for clients to specify relationships between resources and specify representation formats they can render, and for the server to honor these representations or return a response if it cannot (Apigility, 2014).

Web applications rely on data from sources which consist of different structures and formats (Cardoso & Sheth, 2006). Section 2.3 discusses how sources are integrated through web services examining web applications types, web application feeds, application programming interfaces and web apps. Web application software usually appears in a structured hierarchy of directories which contain all resources related to the application such as programming source files and media files (Chaffee, 2000). Each web application typically has a database for storing the data it collects, sends or retrieves. Web application development requires an understanding of heterogeneous systems, programming languages, concepts, and frameworks (Laine, Shestakov, Litvinova & Vuorimaa, 2011). Commonly, web applications are based on a conventional three-tier architecture (Laine et al., 2011). The three-tier architecture is illustrated in Figure 2.1 and consists of the presentation (user interface), logic (server), and data (data management).
tiers which are all authored using conceptually different programming languages (Laine et al., 2011) such as ASP, ColdFusion, Java, PHP, Python or Ruby on Rails (Petersen, 2008). A common example here is a PHP built web application. Its logic tier is a Linux server, its data management tier is the database hosted on the Linux server (MySQL database) and its user interface tier is the front end of the application with which the user interacts.

Figure 2.1 Three tiered client / server

Web applications are complex, ever evolving, and rapidly updated software systems (Mansour & Houri, 2006). To achieve integration of data sources for web applications, some characteristics of file systems and data sources must be considered. Many web applications rely on multiple, heterogeneous data sources (Cardoso & Sheth 2006). A data source or file consists of data and attributes (Bing, Zheng-ding & De-chun, 2001). The application program manipulating files must be able to decode and properly use these files. As data and meta-data are stored together in many files, most files are self-describing (Bing et al., 2001).

Figure 2.2 below uses UML to illustrate a typical web application. The browser sends a request for the web application to the web server. The web server decides what to do with the request. Some elements such as static data sources are returned and presented on the browser. Dynamic data elements are retrieved by querying the application server. The application server passes the request to the correct web application constructing a response using data from the database server. This response is then passed back down the workflow to the web browser (Feiler, 2000; Laine et al., 2011). There are many different types of server environments for web applications.
ranging from open source application servers such as JBOSS, Microsoft Windows based web servers running IIS (Feiler, 2000). Two common examples include PHP application servers which are used for running and managing PHP built applications, and Microsoft’s .NET Framework which are deployed on Windows Servers.

Figure 2.2 UML diagram of web application architecture

Web applications can also be accessed via Internet connected mobile apps on mobile devices (Shuler, 2007). These are commonly referred to as ‘mobile apps’ or just ‘apps’ (American Dialect Society 2010). Mobile apps are designed to run stand alone on a mobile device or in conjunction with a web-based back end, communicating to a server or external website via the Internet (Shuler, 2007). They can then retrieve live feeds, store and re-query user data on demand and present information from a multitude of sources. The Facebook app for example is an iPhone web application that interacts with the Facebook server network, re-querying it to present user’s Facebook feed. It reruns a query underlying the active web page to reflect changes to the records shown to the
user, displaying newly added records. Apps that interact in this manner have characteristics similar to cloud based applications.

Web applications that are hosted in the cloud are described as software with data access and data storage that is within the cloud. The name "cloud" is used because it can "float about" among specific servers, rather than being in a specific place. In fact, the user does not know where the specific server is (Slahor, 2011). A user of cloud computing really does not have to know how all these tasks are done, but, instead, has the cloud "installed" as a system to share in the "pool" of services the cloud offers (Slahor, 2011). In this instance, web applications could be installed on a device pooling any number of resources for data access and data storage and are accessible in real time with 24/7 access. Some common examples include Mozy, Evernote, Sugar Sync, Salesforce, Dropbox, NetSuite, and Zoho.com. Other examples such as web email (Google, Yahoo, Microsoft Hotmail, etc.) depend on cloud technology but are available offline if configured as such (Tajudeen, 2012). As there are many similarities between “cloud” and “web” apps, it is difficult to distinguish between them (Tajudeen, 2012). stemming from the natural similarities that exist between them. But noteworthy differences between the two are seen considering how cloud apps leverage a richer user customization experience and seamless integration with resilient and scalable back-end infrastructure which is characterized by cloud based services (Tajudeen, 2012). An investigation of how they are integrated is examined in the following section.

**2.3 Integrating data sources for use in web-based applications**

**2.3.1 Definition**

In the contemporary business and industrial environment, the variety of data that are used by organisations is increasing rapidly. There is also an increasing demand for accessing this data (Rajeswari & Varughese, 2009). Web applications often have a need for real time retrieval of data from one source presented on another and distribution of data to a range of different sources. To achieve this they require a data integration process. Technologies and approaches exist to achieve this but these cannot be used
as a generic solution in all situations. The reason that this difficult is discussed further in Section 2.4.

Data integration is the process of combining data residing at different sources, and providing the user with a uniform view of these data (Lenzerini, 2008). Integration is the mechanism to search, query or retrieve information from these sources (Offutt, 2002). It is the process of combining software elements, hardware elements, or both into an overall system. It allows for the linking between what a user intends to search, seek or retrieve, and the data or traffic source from which this information is sourced (Lenzerini, 2002). This process is the responsibility of a data integrator, which is a consumer or business aiming to achieve a specific data integration outcome (Wolk, 2006).

Data integration has a number of data sources for storing data and are data services that use the data sources and support business process segments in an enterprise (Nečaský, 2007). Each such service accesses the data from one or more sources, processes it and sends it to its clients in a standard format defined in its schema. A data source may provide data generally in any representation described in the schema. Data sources and services are usually built independent of each other so their structures and/or schemas may be very different to one another (Nečaský, 2007) to achieve different outcomes.

The nature of systems such as news feeds, analytics sources and providers of statistics, all have data integration as a fundamental requirement (Rajeswari & Varughese, 2009). Web services combine data from various sources presenting a consistent view, commonly called a mash-up (Strom, 2008). A web service, as defined by the W3C Web Services Architecture Working Group, is a software application whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts. A web service supports direct interactions with other software agents using XML-based messages exchanged via Internet-based protocols (Haas, 2003). The importance of examining the state of art in data integration illustrates common patterns of challenges being faced in the field. This is an important, yet a very difficult challenge with many information systems (Nečaský, 2007).
Current approaches are often domain or technology specific. Yet there are patterns of common challenges amongst current approaches. Data integration is crucial in large enterprises that own a multitude of data sources for producing data sets that can develop and enhance cooperation among government agencies across World Wide Web (Halevy, Rajaramn & Ordille, 2006).

There are varying types of heterogeneity, those being: System heterogeneity, Structural heterogeneity, Syntactic heterogeneity, Semantic heterogeneity. System heterogeneity occurs due to applications and data residing in different hardware platforms and operating systems (Cardoso & Sheth, 2006). Structural heterogeneity is encountered when different information systems store their data in different document layouts and formats, data models, data structures and schemas. (Cardoso & Sheth, 2006). Some example technologies include XML and XQuery. Syntactic heterogeneity arises when different representations and encodings are used for data. XML supports ability to deal with syntactic heterogeneity (Cardoso & Sheth, 2006). Semantic heterogeneity occurs because the meaning of data may be expressed in different ways. Semantic heterogeneity focuses on the content of an information item and its intended meaning. Sharing and exchanging information in a semantically consistent way is the key factor for successful resolution of this type of heterogeneity (Cardoso & Sheth, 2006).

2.3.2 Web applications types

The web has enabled many different ways to create database applications. A web browser interface allows any user with the proper browser to run the application (Flannery, 2000). This in turn results in an unlimited number of different web application types since practically any idea can be translated into a working web application, each with its own unique custom requirements, presentation and functionality.

To further understand the need for integration of data sources for web applications, the next section presents an overview of the architectural characteristics and some of the web applications that use them which contributes to further understanding the need and purpose of integrating between them. In all of the examples, the web application type is just one of many web application components that exist on these services discussed.
2.3.2.1 Web application feeds

A web feed is a data source used for providing frequently updated content. Really Simple Syndication (RSS) is a family of web feed formats used to publish frequently updated work such as blog entries, news headlines, audio, and video in a standardized format (Libby, 1999; Pilgrim, 2002). It is this format that has become a backbone for live integration between websites. Web feed syndication (mostly known as RSS 2.0) is an underlying technology for social media platforms. When referring to feeds, news feeds, podcasts or videocasts, bookmarks or numerous content sharing tools, these refer to different formats and uses of web feeds (Oleg, 2011).

2.3.2.2 Application programming interface

A web API is typically a defined set of HTTP request messages along with a definition of the structure of response messages, typically expressed in JSON or XML (Benslimane, 2008). Web APIs allow the combination of multiple services into applications which is seen in many different types of web applications ranging from social networks through to custom purpose web software systems. This combination is often referred to as mashups which means software applications that merge separate APIs and data sources into one integrated interface (Zang, Rosson & Nasser, 2008). With the large number of APIs, and the vast and increasing amounts of web content available, mashup technology allows web developers to create a variety of customized, novel web applications (Zang et al., 2008). Effectively, using APIs, data integrators are able to pool together any number of different resources to create a single multi-purpose web application. The following examples illustrate how web applications use APIs to mash different types of web applications into their own web application.

Social networking websites such as Facebook provide an API for developers to add social networking modules to their own websites. A web application known as Pinterest uses the API of Facebook.com to allow new Pinterest members to register on their website using their existing Facebook profile and user account. The API allows for
Pinterest to connect to Facebook and retrieve this user information. Pinterest also uses
the API for finding friends on Facebook and sharing Pinterest features with them by
engaging its web servers to query and retrieve matching friend data.

Flippa.com is a marketplace for buying and selling existing websites and domain names.
Websites that perform as a dynamic web application such as Flippa.com rely heavily on
the APIs of several different resources to present information on its product listings.
Using the APIs of other web resources, they are able to present analytics on website
listings that are put up for sale so that a potential buyer is able to see important statistics
that could affect their buying decision. To compute useful statistics such as Google
ranking, Alexa.com website rank, Google Analytics sales history, Paypal sales history,
number of incoming links and even domain ownership information are pulled together
using APIs from different websites then presented on the Flippa.com listing page.

2.3.2.3 Web apps

The term ‘web apps’ refers to a group of applications that are designed to run on mobile
devices. These apps run through a web browser enabled device such as an iPad,
iPhone or Android based smart phone and can query, retrieve and interact with a web
server like a normal web browser application. These apps can be native or generic
(Brewer, 2011). Native apps are designed and developed for specific mobile operating
systems, such as Symbian, Android, Blackberry and iOS. Each operating system is used
on various mobile devices and can install apps directly onto a device. These native apps
then run using the mobile device's operating system (Brewer, 2011). Generic apps are
those that are designed and developed to function in a web browser, regardless of the
device (Brewer, 2011). However, many native apps have a web-enabled feature which
queries, retrieves and interacts with a web server but are run as a standalone application
on a smart phone without requiring a web browser to use it.

A common example that comes installed on the Apple iPhone is the Weather app. This
is a native app designed to run without the use of a web browser. It does however query
a web server to retrieve in real time the latest information on the current weather
conditions based on the selected city.
An example of a web app that has both native and generic app components is the Commonwealth Bank of Australia app which is a widely used banking app in Australia for one of the country’s four leading banks. The native app components feature various search and information retrieval sections of the app. The web app component launches the Safari web browser on the iPhone to open a window allowing the user to login and check their online banking information.

Each of the above examples both exist as web-based applications as well with the web app servicing as a front end tool for users to engage these web applications exclusively via mobile devices.

2.3.2.4 Summary

The collection of literature discussed suggests that there are three approaches to data integration, models, management process-based approaches and technical approaches which are discussed further in this chapter. A classification scheme is proposed and discussed to demonstrate the applicability of the two approaches for achieving data integration.

2.3.3 Classification

To show how data integration models can be applied to achieve data integration, a classification of data integration models that characterises the design space of each model group is proposed. Researchers often classify competencies based on their logic, theory and purpose (Chyung, Stepich & Cox, 2006; Muhlbacher, Nettekoven & Putnova, 2009). Classifying models for data integration is important to explain the various dimensions of models. Classification schemes assist in recognizing the indicators that models consist of, grouping models by common, repeating characteristics.

As part of classifying data integration models, each model is tested against a list of dimensions. To establish which dimensions to test, existing classification schemes are
examined. Existing classification schemes are based on soft and hard competencies, threshold and performance competencies. They may be organised hierarchically or by prior and empirical classification.

Soft and hard Competencies (Jacobs, 1989) is a classification based on differentiating competencies. Analytical and organisational competencies are considered hard competencies whereas creativity, interpersonal and behavioral skills are soft competencies (Jacobs, 1989). This approach to evaluating the characteristics of web applications is not appropriate as creating soft and hard competencies based on creativity and interpersonal skills are not relevant factors in evaluating the integration suitability of potential technologies. Behavioral skills could be relevant in the context of how an application behaves as a characteristic that is part of a web application focused classification scheme.

Threshold and performance competencies (Boyatzis, 1982) are basic minimum requirement while performance competencies are skills and competencies that actually differentiate between average and excellent performers (Boyatzis, 1982). These competencies could be part of a web application classification scheme in the context of the performance of a web application.

Hierarchical wise classification is a classification based on categorisation of competencies across different hierarchical levels (Lindsay, 1997; Viitla, 2005). Prior and Empirical classifies based on predefined competencies or arrange competencies based on empirical analysis (Boyatzi, 2000). This approach to categorisation could be extended to a web application specific classification scheme.

Existing classification schemes used in the management or transition of data are either too simple or are specific to a technology environment. What is needed is a classification scheme that is at a broader level encompassing a broad range of dimensions and can be applied to groups of models for achieving data integration. There is no standard classification scheme for data integration models. However, there are patterns of common characteristics in data integration models which will form the basis of the classification presented in this chapter. The focus of this chapter is to provide an
analysis suitable for investigating the viability of supporting data integration models in chapter 3.

This section presents five dimensions that describe the hierarchy of different types of models. It builds on the foundation of Hierarchical Wise Classification where by classification is made by categorising the various competencies. Section 2.4 evaluates the model groups and a comparison of their dimensions. In doing so, this lays down the groundwork for the following chapters and enables a discussion on the current approaches to data integration and the need for a more generic top level framework.

The 5 dimensions are as follows:

1. **Data Translation Dimension.** Translation characteristics are important part of data integration because it is about the movement of data from one source to another (Heck, 2000). This is discussed in Goguen, (2005) as important to support translation of data from multiple sources. Section 2.3.3.1 discusses this dimension and examines its process.

2. **Programmable dimension.** This dimension is important because changing the properties, aspects or nature of the data allows for it to be understood by the destination as part of data integration (Dong, Halevy & Yu, 2009). Section 2.3.3.2 discusses this dimension and examines its characteristics.

3. **Centralised dimension.** This dimension is important because of the need to reduce redundancy and streamline the work flows in the data integration process. This is discussed in Shashoua, (2012) and Teradata (2012) as important for web applications for consistency and efficiency. Section 2.3.3.3 examines this dimensions characteristics.

4. **Meta-data dimension.** This dimension is important because whenever data is moved or transformed, meta-data needs to be involved (Bernstein & Bergstraesser, 1999). This is discussed in Bernstein & Melnik, (2007) as important for supporting data mappings for web applications. Section 2.3.3.4 examines the characteristics of this dimension.

5. **Self-manageable dimension.** This dimension is important as it enables the data integration process to be managed by the creator or administrator (Zhao, Chen & Yao, 2008). This is discussed as important in Zorrilla, García, & Álvarez (2010)
due to ensuring control over the data integration process by a non technical expert. Section 2.3.3.5 discusses this dimension and its merits.

**2.3.3.1 Data translation dimension**

This dimension characterises competencies based on how the data is treated between one data source and another. That often involves translating the data (Heck, 2000). Data translation reduces the need for custom programming when combining incompatible data from multiple internal systems, thus enabling scarce IT resources to be used more efficiently and cost-effectively (Heck, 2000).

Different types of translators have been developed. The foundations of data translators stem from two distinct types. String-matching translators operate on tag-based data formats where the translation involves searching for a particular tag in the input format and replacing it with the equivalent tag in the output format (Mamrak, Barnes & O’Connell, 1993). Structure-oriented translators consider the structural constraints of data formats where there may be rules about the structure of certain elements in a document, such as a specific order for certain tags (Mamrak et al., 1993). Both of these translator types are common architectures of automated or domain specific data translators. Most automated translation systems require the developer to manually specify the mappings between different data formats and then automate the other parts of the translation process (Milo & Zohar 1998). It is impossible to automatically determine the correct mapping in all cases (Milo & Zohar, 1998) but using a rule-based system would provide a default set of general rules that attempt to find similarities in the names and structures of constructs across schemas (Milo & Zohar, 1998). This approach is example how data translation can interpret data from one source and map it to another.

Models need to incorporate data translation in their work flows as part of the flow of data in data integration. Figure 2.3 below outlines a high-level view of data translation in the web development process. It shows a UML Component Diagram of how the translation of data sits between the data source and the destination source. The data that comes
from the data source is interpreted by the data translator which then maps the data to targets on the destination source.

Different levels of support for data translation describe the characteristics of four levels of translation. Schema matching tools produce a relation between the node sets of two schemas; nodes may be elements, attributes, paths, or some combination of two or all three of these. Most current tools work at this level, and therefore rely on other tools, perhaps just a text editor, to provide the additional information needed to support data translation and integration (Goguen, 2005). Schema mapping tools provide enough information to generate a view that can be used for data translation. Schema morphism is a mapping that produces correct data translation, noting that the ultimate criterion for correctness is satisfaction of the user. The language of the tool need not be the same as that of the views that it generates; indeed, in the best cases, the tool is GUI-based rather than text based like SQL or XQuery (Goguen, 2005). Data integration tools support translation of data from multiple sources and its subsequent integration to answer queries over a global schema (Goguen, 2005). Heterogeneous data integration tools
provide translation and integration for data sources over schemas of different species (Goguen, 2005).

When the sources are of dissimilar types, such integration can be made more difficult in that the data may be different in their internal representation. When this is the case, the data must be translated to the format of the destination to be usable by that source (Burris, 2001).

It is for these reasons that data translation is crucial as one of the dimensions to explore in the design space of models for data integration. Data translation is commonly found in Language models, Layer models and Schema mappings and will be one of the benchmarks for comparison by classification scheme.

2.3.3.2 Programmable dimension

The programmable dimension characterises competencies based on how easy it is to manipulate and define the properties or controls of the data as it is altered and manipulated throughout the workflows of the data integration model.

An example manipulation of the data is the ability to reformulate a data query. Typical behaviour here includes enabling the data integration to reformulate a query posed (Halevy, 2001). The system needs to translate these queries into some structured form so they can be reformulated with respect to the data sources (Dong et al., 2009).

The ability to facilitate alteration of the data properties is an important design element to ensuring successful data integration. That is why this element forms one of the dimensions to test the design space of models for data integration.

Figure 2.4 shows a UML Component Diagram illustrating an example where a model would enable the manipulation of the properties or controls of the data. This manipulation allows for an intervention to occur during the integration process to shape, transition or manipulate the outcome of the data integration process and determine how the data ends up.
Figure 2.4 shows in simple terms how the programmable dimension sits within the data flow in data integration. The manipulation of the data occurs as the data leaves its source and before it reaches its destination. It is at this point that the data can be altered and manipulated to be understood at its destination.

It is for these reasons, the programmable dimension is crucial as being one of the dimensions to explore the design space of models for data integration. The programmable dimension is commonly found in models such as Data Mapping, Language Models and Schema Mappings and will be one of the benchmarks for comparison by classification scheme of different groups of models for data integration.

2.3.3.3 Centralised dimension
This dimension characterises competencies based on their structure so that the integration process is dependent on a centralised platform to eliminate redundant hardware and software and the associated support and maintenance costs (Teradata 2012).

The behaviour expected consists of ensuring that all the data has been mapped before it moves into a centralised location to view (Shashoua, 2012). An example of the centralisation of data is where all the integration work flows made are transmitted to the central processor to update the integration results (Hong, 1995).

A model that is centralised allows for the data integration process to be administered and deployed without the reliance on a technical expert. The centralised dimension sits within the data flow in data integration. Data flows are mapped to a centralised platform before reaching the destination.

This is important for model approaches to indicate models that have centralised characteristics. Models that are centralised could indicate they are more flexible in nature, less redundancy, support and maintenance.

It is for these reasons, the Centralised dimension is crucial as being one of the dimensions to explore the design space of models for data integration. The Centralised dimension is commonly found in models such as Schema Mappings and Common Models and will be one of the benchmarks for comparison by classification scheme of different groups of models for data integration.

### 2.3.3.4 Meta-data dimension

This dimension characterises competencies based a tagging method to describe the elements of data in a data integration process. These tags describe the characteristics and components of fields, files, views, elements and structures of the data. Meta-data controls the flow by specifying such information as location, schedule, and source-to-target mappings (Bernstein & Bergstraesser, 1999).
Meta-data refers to data that describes data. Meta-data is an integral element in model approaches to data integration. It enables the data destination to understand what the data looks like, what it is receiving and where to place it. Model approaches that have meta-data characteristics could indicate a higher level of flexibility for identifying the data elements being integrated with. By tagging the data constructs, the target data source can identify what data belongs where as part of its integration process.

The work flow between the data source and the destination consists of a tag to describe the data element being integrated. This tag describes what the data looks like. This enables the destination to understand what the data is its receiving and where to place it. Like any kind of data, meta-data requires a persistent data store such as a repository (Blain & Elkington, 1996). It also requires an information model (i.e., schema) that describes the meta-data to be managed (Blain & Elkington, 1996).

It is for these reasons, the Meta-data dimension is crucial as being one of the dimensions to explore the design space of models for data integration. The Meta-data dimension is commonly found in models such as Meta-Tagging Models and Schema mapping models and will be one of the dimensions of classification in the proposed scheme.

### 2.3.3.5 Self-manageable dimension

This dimension characterises competencies based on enabling the data integration process to be managed by the creator or administrator. Many models however require a technical writer to create the data integration model but self-manageable models would mean anyone who does not have the skill set of a technical writer can implement the model and its processes without the need for a technical expert.

In this dimension the control and ownership of the data within the data integration process can be managed by a non-technical user. An Information Technology (IT) expert does not have to manage the data integration process, minimizing deployment, administration and maintenance costs. Practically, user requirements and preferences decide strategies of abnormal situation handling, and explanations of mined patterns.
(Zhao, Chen & Yao, 2008). In other words, the user requirements they seek as part of the data integration process will determine the outcomes directly rather than being translated to a more technical control of the work flow, outside the control of the user.

Model approaches that are self-manageable are important because they empower the data integrator to have control over the management of their data integration process. A model that is self-manageable allows for the data integration process to be administered and deployed without the reliance on a technical expert as detailed in workflow. IT involvement sits outside the data flow and there is no link between the two. The deployment and administration can be accomplished outside of IT involvement.

A model in this type enables the creator or administrator to manage all aspects of the model on an ongoing basis as requirements change once the initial data integration process has been implemented or deployed. In contrast, some models require a technical writer to create the data integration model but any one beyond that can implement the model and its processes without the need for a technical expert.

It is for these reasons, the self-manageable dimension is crucial as being one of the dimensions to explore the design space of models for data integration. This dimension will be one of the benchmarks for comparison by classification scheme of different groups of models for data integration.

The dimensions will be used in an analysis of the model, technical and management process-based groups discussed further in this chapter to compare the characteristics of each group. In doing so, this lays down the groundwork for the following chapters and enables a discussion on the current approaches to data integration and the need for a more generic top level framework. The five dimensions listed are enough to measure these characteristics without the need for further dimensions to be examined. The dimensions measure translation characteristics for the data, the programmable characteristics, if the web application is centralised or decentralised, meta-data and if it is self-manageable. Different spectrums of a web applications can be contrasted using the same list of dimensions and enough characteristics of a web application can be examined using the dimensions list. For example, a custom built online sign up web application that is part of a public facing website where users can sign up via an online
form including entering credit card payment details for a subscription based service does involve translation of data due to the information being posted from the web form to both a payment service and the server. It is programmable, it is centralised, it involves writing meta-data for the public facing website and the web services but it does have a reliance on a IT expert to maintain due to the payment interface and custom built form.

2.3.3.6 Comparison of model groups

These criteria act as a means to review each of the model categories ability to support data integration. Table 2.1 summarises these dimensions that each model will be tested against:

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Dimension Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Translate</td>
<td>Does the model translate the data?</td>
</tr>
<tr>
<td>(2) Programmable</td>
<td>Is it programmable or is it fixed?</td>
</tr>
<tr>
<td>(3) Centralised</td>
<td>Is it Centralised or deCentralised?</td>
</tr>
<tr>
<td>(4) Meta-data</td>
<td>Does it involve writing meta-data?</td>
</tr>
<tr>
<td>(5) Self-manageable</td>
<td>Is it self-manageable?</td>
</tr>
</tbody>
</table>

Table 2.1 Classification Scheme Dimensions for Data Integration

The following sections provide an analysis of model types classified under different categories in terms of their characteristics and dimensions.

2.4 Data integration models

2.4.1 Overview

Data Integration models support data integration process. They do that by clearly modeling the data processes and flows required to achieve data Integration. They are a
conceptual abstract plan referring to the overall data flow structure or type (Kay, 2003). A model is an artifact formulated in a modeling language, such as Unified Modeling Language describing a system through the help of various diagram types (Rumbaugh, Jacobson & Booch, 1999). Models are used either as a source to aid data integration or as an implementation tool. Models for data integration can be structural or semi-structural and can exhibit many complex data relationships and can be issued by many formats in favor of data exchange (Jinan & Guojing, 2010). There are different classifications of models such as data mapping, schema, translators, common and domain specific. The complexity of mashups for example is a major challenge for data integration and so various models exist to create an easier navigation for users when dealing with mashup data sets.

Collectively, there is a multitude of existing and developing models, methodologies, applications and languages for achieving data integration and each retains its validity and desirability in its respective application realm. Model groups such as layer models, common, schema, data mapping and language models all boast strong support for data integration. However, many of these are intended to support integration within a specific domain.

This section will focus on reviewing the literature around defining the types of model approaches discussing the complexity, heterogeneity, the process and sources. The classification scheme introduced in section 2.3.3 is discussed to demonstrate the applicability of the different model approaches for achieving data integration.

2.4.2 Common data models

Common Data Models have been related to standardization, an approach to avoid costly customised programming as companies trade data and feed it into their systems (McCarthy, 1998). A Common Data Model (sometimes referred to as a Canonical Data Model, or Common Model in short) is an application independent data model describing the structure and data semantics in relation to an organisations business processes (McCarthy, 1998).
Common models are for management of information for systems, networks, applications and services, and they allow for vendor extensions. They provide common definitions that enable vendors to exchange semantically rich management information between systems throughout the network (McCarthy, 1998). They are composed of a Specification and a Schema. The Schema provides the actual model descriptions, while the Specification defines the details for integration with other management models.

The benefits of common models include that an organisation would have a data model that contains descriptions of all the data used by the organisation (McCarthy, Shawn 1998). It also allows for consistent management of data elements, independent of their manufacturer or provider (McCarthy, 1998). The model is intended to provide a framework for data integration of each data element by different users / organisations regardless of their end IT environments (McCarthy, 1998). It provides a way to control and manage these data elements (McCarthy, 1998). It is intended that software can be written once and work with many implementations of the common model without complex and costly conversion operations or loss of information (McCarthy, 1998).

An example Common Data Model is the Unidata Common Data Model which merges the Network Common Data Form (NetCDF), the protocol OPeNDAP, and the data model HDF5 to create a common API for many types of scientific data (Nativi, 2008). NetCDF is a set of software libraries and data formats that support creating, accessing, and sharing of array-oriented scientific data (Unidata, 2017). OPeNDAP is a framework that simplifies all aspects of scientific data networking (OPeNDAP, 2017). HDF5 is a library that provides a programming interface for implementing a model for managing and storing data (HDF Group, 2017). A Common Data Model provides common definitions that enable vendors to exchange semantically rich management information between systems throughout the network. Common data models are usually self-describing and based on some kind of graph model in which both schema and data are stored together (Bing et al., 2001). Figure 2.5 shows how the Common data model is composed of a Specification and a Schema with the Schema providing the actual model descriptions, while the Specification defines the details for integration with other management models. It shows the relationship of the Common Data Model between data sources, rules, schemas and classes. It shows the two way connection between a data source and its rules, schema and classes which define the data source and its structure.
The Common Information Model has been categorised as both costly to develop and maintain as well as developed too quickly by the vendors who create them (McCarthy, 1998; Wyss & Robertson, 2005). Rather than focusing on the quality for which its target users rely on them for, vendors value speed over quality when creating a Common Information Model (McCarthy, 1998; Vosgien, 2015). A negative outcome can be stagnation (McCarthy, 1998) and reliance on a technical expert to implement (Wyss & Robertson, 2005). This could also lead to a lack of support for new and emerging technologies since the goals of a Common Information Model that is developed quickly would only address the current needs and requirements rather than consideration for new and emerging requirements. Because of the length of time involved in reaching agreement over a common data definitions (Tierney, Nolan, Robinson, & Armstrong, 2014), many systems and ideas can stagnate. By the time common data definitions have been agreed upon, technologies and system requirements are out of date (Clark, Krumm & Bieleski, 1992; Gujarathi 2011; Wyss & Robertson, 2005).

The common data model acts as a kernel data structure connecting all application programs, supported with file and data management software modules that control the user interface as well as the information security and integrity (Gujarathi, 2011). It is a
collection of parameters which, in turn, forms all the features in the design (Gujarathi, 2011). The common data model needs to be associated and supported with programs to keep its structure and contents consistent and updated. The application of the common data model requires standards and governing codes (Gujarathi, 2011). Any change in any one of the design parameters had an impact on rest of the design. A major limitation of the common data model in this case was that implementations without established design procedures that had ad hoc user interventions and rolling backward or forward meant there was a lack of the flexibility with this model (Gujarathi, 2011).

Another shortcoming of the common information model is an overreaction to OSI management (McCarthy, 1998). Historically the ISO and ITU-T standardised very slowly in the management field and standardisation cycles took four years (McCarthy, 1998). They conclude that the way management information is currently modeled by the IETF is a reaction to the slow pace of OSI management information modeling from its origins. Furthermore the lack of researcher driven Common Information Models suggests that the best technology experts are rarely involved in standardization suggesting that researchers intend to do things right rather than fast which competes with the observed fast over quality vendor driven approach.

Given the architecture described on how common models are structured, models in this group do allow for the translation of data in their work flow. As the data moves between the source and destination, the translation occurs at this point. The common data model provides instruction for how this translation occurs.

Common data models do not contain programmable elements (Gujarathi, 2011). Models in this group provide a set of definitions for defining the workflow of an integration process, but not physical programmable boundaries that can be manipulated and updated based on a programming language.

Common models are centralised (McCarthy, 1998) as they encourage a single defining element to manage the integration process. The model sits between the source and the destination in the integration work flow where it acts as a single defining platform.
There are meta-data elements within common models, descriptions can be short sentence or simple text structured in nature (McCarth, 1998; Gujarathi, 2011) but expressed with rich meta-data elements such as technical meta-data (Estlund, Fleming, Matienzo, Stroop, 2015). Common data models are not self-manageable, they are typically written by an IT expert and maintained by such (McCarthy, 1998; Wyss & Robertson, 2005) with a lack of flexibility (Gujarathi, 2011).

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Dimension Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Translate</td>
<td>Yes</td>
</tr>
<tr>
<td>(2) Programmable</td>
<td>No</td>
</tr>
<tr>
<td>(3) Centralised</td>
<td>Yes</td>
</tr>
<tr>
<td>(4) Meta-data</td>
<td>Yes</td>
</tr>
<tr>
<td>(5) Self-manageable</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 2.2 Common data model work flow dimensions

Table 2.2 illustrates the mapping of the Common Data Model Work Flow to the classification introduced in section 2.3.3

2.4.3 Data mapping

Every business system, whether it manages sales orders, purchase orders, inventory, payables or receivables, relies on information derived from other systems (Kazzaz, 1997). Data mapping is a process that creates a relationship between two databases aiming to solve the problem of integrating two sets of system data where these systems have their own unique format and structure (Kazzaz, 1997). Data mapping describes structural representations of data. Data mapping is fundamental in data cleaning, data integration and semantic integration and are the foundation for constructing large-scale semantic web information systems which facilitate cooperation of autonomous structured data sources (Fletcher, 2007).
Mapping is a system of defined pairs which determine how the Data Service and Data Source classes and attributes are implemented. Mapping is generally only one-way. Mappings are the basic glue for constructing large-scale semantic web information systems which facilitate cooperation of autonomous structured data sources (Fletcher, 2007). To map, you must know how the tables and columns of one database (Database A) relate to the tables and columns of another database (Database B) in a relational model. In a graph model such as resource description frameworks (RDF) graphs, the representation of requirements and specification of design component properties is done via graphing these relationships (Nassar, Austin 2013). The core of the mapping solution is establishing a simple cross-reference that electronically "maps out" this relationship (Kazzaz, 1997).

As part of data integration, designing the relationships between two data sources is called data mapping (Alexe, 2008). Figure 2.6 illustrates a mapping relationship between the data source and its matching attribute. This relationship contains parameters describing the elements flowing from the data source to match up to an attribute.

Consider this example:

Database A: has a customer table which is structured with a customer reference number field, customer name, customer address and suburb field.
Database B: has a purchase table which is structured with a purchase id, customer number, purchaser name, purchaser address and location.

These fields can be mapped between the two databases. Figure 2.7 shows a UML Component Diagram mapping the relationship between two different databases with a connection between the fields of Database A mapped to correlating fields in Database B of a similar nature.

![Database Mapping Diagram](image.png)

Figure 2.7: An example data mapping scenario

A possible negative outcome of implementing data mapping is that data definition agreements are required between all parties. Creating a single consistent data definition can be challenging when negotiating between the various parties involved. This can be a lengthy process which will lead to many additional side effects, such as the need to update documentation.

In summary, data mapping models establish a relationship between data sources in a typically one way manner. Data mapping models allow for the translation of data in their
work flow. As the data is mapped from the source, the translation occurs to be understood by its mapped attribute. Data mapping has been described as consisting of problematic characteristics within schema matching, schema mapping and model matching.

Mapping models contain programmable elements. The attributes within this model are entirely programmable to manipulate the data between the source and the destination. These elements allow for technical alteration to receive data fields being passed during the integration process (Fletcher, 2007; Alexe, 2008).

Models in this group are not centralised as they do not have a single platform for managing the integration process as described in Figure 2.10 attribute mapping model example. There are no meta-data elements within mapping models, descriptions are instead limited to the mapping attributes themselves (Fletcher, 2007; Alexe, 2008). Mapping models are also not self-manageable because they are typically written by and controlled by an IT expert (Fletcher, 2007).

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Dimension Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Translate</td>
<td>Yes</td>
</tr>
<tr>
<td>(2) Programmable</td>
<td>Yes</td>
</tr>
<tr>
<td>(3) Centralised</td>
<td>No</td>
</tr>
<tr>
<td>(4) Meta-data</td>
<td>No</td>
</tr>
<tr>
<td>(5) Self-manageable</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 2.3 Compares this model with the dimensions of the classification scheme

2.4.4 Language models

Language models are based on a specific query language as an approach to developing either data mappings or schema mappings to support data integration. They are programmable in nature (Wyss & Robertson, 2005).
Many language models have been devised in an attempt to create improved querying of data sources such as K*SQL. It is a data model and language which generalizes both words and tree structures extending on relational query languages (Mozafari, 2010). Query languages have been used as a high-level uniform approach for schema mappings and have elements to it that could be applied to data integration too (Marnette & Papotti, 2010).

Language models are classed as fit for a specific purpose. A language model developed called FIRA (Wyss & Robertson, 2005) has been created to act as a medium for data integration. It can assist with the creation and maintenance of wrappers, mediating functions, and mediated schemas, especially in the case of relational data sources (Wyss & Robertson, 2005).

Language models face problems with missing or incomplete information (Wyss & Robertson, 2005). They are often domain specific designed to support a specific purpose. This could lead to a lack of support for new and emerging technologies. Uncertain or imprecise data are pervasive in applications like location-based services, sensor monitoring, and data collection and integration (Xie, 2008). For these applications probabilistic databases can be used to store uncertain data, and querying facilities are provided to yield answers with statistical confidence (Xie, 2008). This approach to achieve and maintain data quality during data integration at the transportation stage demonstrates how query answers can be improved.

In summary, language models are specific in nature and often developed for a unique data integration environment or purpose. This could lead to a lack of support for new and emerging technologies since the scope of a language model is aimed at a specific technology and purpose.

Models in this group allow for the translation of data in their work flow. As the data is queried between the source and destination, the translation occurs at this point. Language models provide instruction for how this translation occurs at this point.

Language models contain programmable elements (Wyss & Robertson, 2005). Models in this group provide a set of definitions via query language for defining the workflow of
an integration process based on physical programmable boundaries that can be manipulated and updated based on a programming language (Wyss & Robertson, 2005).

Language models are not centralised as they do not encourage a single defining element to manage the integration process (Wyss & Robertson, 2005). They instead are diverse in structure and may have a multitude of integration elements compared to a centralised platform.

Meta-data elements are not common within language models, descriptions are instead programmable in language and are query structured (Wyss & Robertson, 2005). Language models are not self-manageable as they are reliant on an IT expert to create and maintain on an ongoing basis due to the programming experience required to implement language models (Wyss & Robertson, 2005).

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Dimension Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Translate</td>
<td>Yes</td>
</tr>
<tr>
<td>(2) Programmable</td>
<td>Yes</td>
</tr>
<tr>
<td>(3) Centralised</td>
<td>No</td>
</tr>
<tr>
<td>(4) Meta-data</td>
<td>No</td>
</tr>
<tr>
<td>(5) Self-manageable</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 2.4 Language model dimensions

2.4.5 Schema Mappings

Schema mapping is the process of converting an instance of one schema, called the source schema, into an instance of a different schema, called the target schema (Marnette & Papotti, 2010). Schema mappings models aim to improve data exchange between different sources (Marnette, 2010). Schema mappings are expressions that specify how an instance of a source database can be transformed into an instance of a target database. In this model, challenges occur when transforming data from one
source to another and it is not always possible to achieve this transformation in the first place (Marnette, 2010).

Defining an XML schema up front is ultimately fruitless as it will need to change over time (Wolk, 2006). If the schema, the shared store that supports them and every system that uses them has to move to a new version all at once, one cannot achieve data integration successfully. Systems could have to wait for necessary changes, because other systems are not at a point where they can adopt a revision. Systems could be forced to do extra, unexpected work, because other systems need to adopt a new revision (Wolk, 2006). To solve this problem, a versioning strategy allows the schema and store to move ahead independent of the rate at which other systems adopt their revisions.

Schema mappings improve data exchange between schemas (Marnette & Papotti, 2010). Figure 2.8 shows a UML Component Diagram illustrating this relationship. The dominant architectural model today is federated, where sources export (import) views to (from) a mediated schema (Wyss & Robertson, 2005). This is a relational data model that requires both wrappers encapsulating source data repositories as well as mapping functions giving the translation between source data and/or schemas and the mediated schema.
Common challenges are experienced with transforming and integrating meta-data between schemas resulting in compatibility issues or obstacles in achieving schema mapping (Marnette & Papotti, 2010). Meta-data reflected in schemas is typically insufficient to assess the data quality of a source especially if only a few integrity constraints are enforced (Rahm, 2000).

Another challenge is in the description of schema elements leading to interpretation shortcomings. It is argued that the fundamental difficulty is to infer real-world semantics of data from the syntactical clues in their representations or their values (Guilian, 2006) because these clues, including schema element names, types, local structures, constraints and value patterns, are often ambiguous, even misleading, and unreliable. There is also research that concludes more support is needed for schema and data transformations rather than a focus just on data or schema meta-data level (Rahm, 2000).
XML will make some things easier, but cannot solve the basic problems, which are not only technical, but also social, as shown by difficulties with implementing the semantic web vision (Berners-Lee, Hendler & Lassila, 2001), as well as workflows which automate processing the enormous datasets that are increasingly common in astrophysics, proteomics, high energy physics, ecology, agriculture, pharmacology, e-business, geology, and many other areas (Goguen, 2005).

Schema mapping models allow for the translation of data in their work flow. As the data is mapped from the source, the translation occurs to be understood by its mapped schema as described in the schema breakdown model Figure 2.10.

Schema mapping models contain programmable elements within each schema (Wolk, 2000; Berners-Lee et al., 2001). The schemas are entirely programmable to manipulate the data between the source and the destination. These elements allow for technical alteration to receive data fields being passed during the integration process.

Models in this group are centralised as there is the ability to have a single platform for managing the integration process as part of the integration as described in the schema breakdown model Figure 2.10.

Meta-data is a core ingredient in the structure of schema mapping models. Schema mapping models use meta-data to describe each of the field attributes (Wolk 2006; Berners-Lee et al., 2001). This supports the work flow between data sources and their destination so that the data can be mapped to the right destination based on its matching meta-data description.

Schema mapping models are not self-manageable, they are typically written by and controlled by an IT expert due to the reliance on programming experience to implement them (Wolk, 2006; Berners-Lee, Hendler & Lassila, 2001). Table 2.5 summarises these comparison discussions against the classification dimensions.
<table>
<thead>
<tr>
<th>Dimension</th>
<th>Dimension Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Translate</td>
<td>Yes</td>
</tr>
<tr>
<td>(2) Programmable</td>
<td>Yes</td>
</tr>
<tr>
<td>(3) Centralised</td>
<td>Yes</td>
</tr>
<tr>
<td>(4) Meta-data</td>
<td>Yes</td>
</tr>
<tr>
<td>(5) Self-manageable</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 2.5 Schema mappings dimensions

2.4.6 Meta-tagging models

Meta-data is structured information that describes, explains, locates, or otherwise makes it easier to retrieve, use, or manage an information resource. Meta-data is often called data about data or information about information (Hodge, 2001). Meta-tagging models consist of tagging approaches to describe the data elements moving from the data source to the destination. Models that have meta-tagging characteristics resemble structures that are made up description hierarchies (Kühne, 2005). The descriptive tags typically consist of a description and keywords describing the data element or attribute called Meta-tags. Models in this group detail several approaches to real time data meta-tagging which are the table approach and categorisation approach.

Figure 2.9 shows a UML Component Diagram illustrating the tagging model workflow. The meta-tag element describes key parameters of integrated elements. Typically this consists of a description and keywords. The description might consist of a short sentence describing the data element or attribute consists of with keywords to describe it as well.
There are three main types of meta-data that describe the nature of how meta-tagging is used in models. Descriptive meta-data describes a resource for purposes such as discovery and identification. It can include elements such as title, abstract, author, and keywords (Hodge, 2001). Structural meta-data indicates how compound objects are put together, for example, how pages are ordered to form chapters (Hodge, 2001). Administrative meta-data provides information to help manage a resource, such as when and how it was created, file type and other technical information, and who can access it (Hodge, 2001).

These different areas of meta-data are used in model groups to describe the links between different work flows between sources and destinations. There are many different ways meta-tagging is applied. The table approach is to build a separate global table in the database management system where all tags are maintained. For each tag the links to the appropriate streaming data elements in the form of query are stored, as illustrated in Table 2.6.
<table>
<thead>
<tr>
<th>Tag</th>
<th>Link to data</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Running”</td>
<td>SELECT measurement FROM HeartRate WHERE time &gt; 9:00AM and time &lt; 9:30A</td>
</tr>
</tbody>
</table>

Table 2.6 Tagging approach

In this approach the tags are maintained separately from the streaming data. As a result, this method may potentially incur significant data overheads since all tags arriving to database separately from the data must be processed, and for every tag an entry in the central tag table must be created or updated.

Four elements to real time data stream meta-tagging categorise what data can be retrieved and in what state of a data stream. The elements are show in Table 2.7.

<table>
<thead>
<tr>
<th>Stream Data Tracking</th>
<th>Tagged objects can be located and tracked with no uncertainty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creation of rich user profiles</td>
<td>Information about user’s interests, mood, observations, and character can be revealed based on the tags in real time</td>
</tr>
<tr>
<td>Exploration and browsing of data</td>
<td>Exploiting tags as a navigation mechanism allowing users to find related streaming data based on the tags</td>
</tr>
<tr>
<td>Social communication</td>
<td>Allowing other people to tag a specific subset of real-time data with their own tags ultimately to see what different people think about the same piece of information</td>
</tr>
</tbody>
</table>

Table 2.7 Categorisation of data stream elements

In summary, models for meta-tagging are a data rich approach to integrating with the data that surrounds data streams and could be applied to many other applications as well. However, it is an intensive exercise with scalability considerations, output rate, latency and resource challenges.
Meta-tagging data models contain programmable elements (Hodge, 2001; Kühne, 2005) as the meta-tags are a set of definitions for defining the data and its attributes. Meta-tags are written in a programming style language.

Meta-tagging models are not centralised as they do not encourage a single defining element to manage the integration process. They instead are diverse in structure and may have a multitude of integration elements compared to a centralised platform (Hodge, 2001; Kühne, 2005).

Meta-data is a core ingredient in the structure of Meta-tagging models (Hodge, 2001; Kühne, 2005). Models in this group use meta-data to provide descriptions of the data being integrated with through keywords and short tagging descriptions. This supports the work flow between data sources and their destination so that the data can be mapped to the right destination based on its matching meta-data description.

Models in this group are not self-manageable, they are typically written by and controlled by an IT expert due to the technical experience required in implementing this model group (Hodge, 2001; Kühne, 2005).

Table 2.8 compares this model type with the dimensions of the classification scheme. It summarises the comparison discussions against the classification dimensions.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Dimension Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Translate</td>
<td>Yes</td>
</tr>
<tr>
<td>(2) Programmable</td>
<td>Yes</td>
</tr>
<tr>
<td>(3) Centralised</td>
<td>No</td>
</tr>
<tr>
<td>(4) Meta-data</td>
<td>Yes</td>
</tr>
<tr>
<td>(5) Self-manageable</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 2.8 Data stream meta-tagging model dimensions
2.4.7 Layer models

Layer models involve a framework of layers, each a different step in the data integration process. Layer models support data integration by sitting at the high-level of the integration workflow as an architecture to guide how the integration occurs.

An example layer model is the Open System Interconnection (OSI) model defines a networking framework for implementing protocols in seven layers. Control is passed from one layer to the next, starting at the application layer in one station, proceeding to the bottom layer, over the channel to the next station and back up the hierarchy according to OSI (2010).

Each layer is defined as follows:

Application (Layer 7)
This layer supports application and end-user processes. Communication partners are identified, quality of service is identified, user authentication and privacy are considered, and any constraints on data syntax are identified. Everything at this layer is application-specific. This layer provides application services for file transfers, e-mail, and other network software services. Telnet and FTP are applications that exist entirely in the application level. Tiered application architectures are part of this layer (OSI, 2010).

Presentation (Layer 6)
This layer provides independence from differences in data representation (e.g., encryption) by translating from application to network format, and vice versa. The presentation layer works to transform data into the form that the application layer can accept. This layer formats and encrypts data to be sent across a network, providing freedom from compatibility problems. It is sometimes called the syntax layer (OSI, 2010).

Session (Layer 5)
This layer establishes, manages and terminates connections between applications. The session layer sets up, coordinates, and terminates conversations, exchanges, and
dialogues between the applications at each end. It deals with session and connection coordination (OSI, 2010).

Transport (Layer 4)
This layer provides transparent transfer of data between end systems, or hosts, and is responsible for end-to-end error recovery and flow control. It ensures complete data transfer (OSI, 2010).

Network (Layer 3)
This layer provides switching and routing technologies, creating logical paths, known as virtual circuits, for transmitting data from node to node. Routing and forwarding are functions of this layer, as well as addressing, internetworking, error handling, congestion control and packet sequencing (OSI, 2010).

Data Link (Layer 2)
At this layer, data packets are encoded and decoded into bits. It furnishes transmission protocol knowledge and management and handles errors in the physical layer, flow control and frame synchronization. The data link layer is divided into two sub layers: The Media Access Control (MAC) layer and the Logical Link Control (LLC) layer. The MAC sub layer controls how a computer on the network gains access to the data and permission to transmit it. The LLC layer controls frame synchronization, flow control and error checking (OSI, 2010).

Physical (Layer 1)
This layer conveys the bit stream, electrical impulse, light or radio signal through the network at the electrical and mechanical level. It provides the hardware means of sending and receiving data on a carrier, including defining cables, cards and physical aspects. Fast Ethernet, RS232, and ATM are protocols with physical layer components (OSI, 2010).

Figure 2.13 illustrates where models in this group would sit in a data integration process:
Translation of data occurs during work flow between the source and the destination via the different layers. The data work flow is structured based on the layers and their boundaries as it is delivered to the destination. The layer model describes how the translation occurs.

Layer models do not contain programmable elements. Models in this group provide a high-level view of how the integration work flow will be structured rather than facilitate the physical elements and attributes of the data integration.

Models in this group are centralised as there is the ability to have a single platform for managing the integration process as part of the integration between the source and the destination via the schema.

There are no meta-data elements within language models, descriptions and behavior is instead structured via the models different layers. Layer models are not self-manageable, they are reliant on an IT expert to create and maintain layer model structures on an ongoing basis.
Table 2.9 summarises these comparison discussions against the classification dimensions. It compares this model type with the dimensions of the classification scheme.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Dimension Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Translate</td>
<td>Yes</td>
</tr>
<tr>
<td>(2) Programmable</td>
<td>No</td>
</tr>
<tr>
<td>(3) Centralised</td>
<td>Yes</td>
</tr>
<tr>
<td>(4) Meta-data</td>
<td>No</td>
</tr>
<tr>
<td>(5) Self-manageable</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 2.9 Layer models dimensions

2.4.8 Other models

Models outside the scope of the model categories discussed have similar aims but are standalone through various specific objectives. A model to deal with erroneous values from data sources prevents correct linking between one source and another (Guo & Dong, 2010). The model also aims to address real world exceptions to the uniqueness constraints of a database system and situations where enforcing uniqueness can miss correct values. Finally, locally resolving conflicts for linked records may overlook some areas. Service and policy based management are still in their infancy with standards still being defined (Martin-Flatin, 2001).

There are no standards today for querying streams as each system has its own semantics and syntax (Zaniolo, 2010). Without a clear understanding of features and semantics, applications are not portable and can be hard to build (Zaniolo, 2010). To overcome this, a descriptive model allows users to examine the behavior of systems and understand the results of window-based queries for a broad range of heterogeneous sources. This model focuses on an analysis of behavior rather than achieving data integration but its root challenges are the same that a data integrator would face when working with querying streams.
Models such as the description model and erroneous values model, that do not fall into a specific category of models, still contain characteristics that are consistent with the classification scheme of being translatable, programmable, centralised, contain metadata elements or aren't self-manageable.

2.4.9 Analysis

Each category of models has its application and benefits but also has various difficulties associated with data integration. Collectively, there is a multitude of existing and currently in development models, methodologies, applications and languages for achieving data integration and each retains its validity and desirability in its respective application realm.

Models groups such as layer models, common, schema, data mapping and language models all boast strong support for data integration. However, many of these are intended to support integration within a specific domain. The classification scheme discussed examined the applicability of the model groups for achieving data integration by comparing the characteristics of each model group against the dimensions of the classification scheme to aid in the understanding of different model types for data integration.

Table 2.10 illustrates the different models in comparison to each other against the criteria discussed.

<table>
<thead>
<tr>
<th>Model Type</th>
<th>Translation</th>
<th>Programmable or Fixed</th>
<th>Centralised or DeCentralised</th>
<th>Meta-data</th>
<th>Self Manageable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Data Model</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Data Mapping</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Language</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Models</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>----</td>
</tr>
<tr>
<td>Schema Mappings</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Meta-Tagging Model</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Layer Models</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 2.10 Model comparison against classification scheme

In Table 2.10, the different model groups are grouped against the different dimensions of the classification scheme with their characteristics compared against each dimension.

Given the architecture described on each model group and how they are structured, models in all groups do allow for the translation of data in their work flow. As the data moves between the source and destination, the translation occurs at this point. Each of the models provides instruction for how this translation occurs.

Common data models and Layer Models do not contain programmable elements whilst the rest of the model groups do. Common models and Schema Mappings are centralised as they encourage a single defining element to manage the integration process. These model groups sit in the middle of the source and the destination in the integration work flow acting as a single defining platform to determine how the integration works and defines the integration elements. On the other hand Data Mapping, Language Models and Meta-tagging Models where they are not centralised.

There are no meta-data elements within data mapping, layer models and language models. Descriptions are instead short sentence or simple text structured in nature. However Schema Mapping, Common and Meta-tagging Model groups do have meta-data elements.
None of the model groups are self-manageable as they are typically written by an IT expert and maintained by such. There are a variety of the challenges that are evident from the different groups of models when applying them to a data integration process.

One of the key problems is the development in time. The data structure and operations on data are changing. In a complex environment, the problem is more difficult as the applications use many data sources (Klimek & Nečaský, 2010). Some systems will have to wait for necessary changes, because other systems are not at a point where they can adopt a revision. Some systems will be forced to do extra, unexpected work, because other systems need to adopt a new revision. To solve this problem, there is a need to embrace a versioning strategy that allows the schema and store to move ahead independent of the rate at which other systems adopt their revisions.

Meta-data, schema mappings, tagging models and the creation and maintenance of layer and common models all require a high-level IT involvement for the creation and ongoing maintenance of these supporting models. They require reliance on a technical expert to change or adapt the model as the nature of the data integration flow changes. The moment the data becomes unexpected or evolves beyond the original scope of the model, these model groups become non self-manageable. This leads to the lack of support for new and emerging technologies.

There are several common causes of failure in data integration projects reported in the literature such as demanding too much information, no effective versioning strategy, and no support for system-level extensions (Wolk, 2006). Approaches include making schema elements optional and encoding system specific occurrence requirements as part of that system's implementation (Wolk, 2006). Another approach is developing systems that produce data according to their version of a shared schema but allow systems to adopt schema revisions at different rates without changing the schema namespace (Wolk, 2006). Another is allowing systems to extend shared schemas with their own data to meet new requirements independent of data-model revisions (Wolk, 2006). The implementation of these solutions is domain specific, requires reliance on a technical expert and requires a great deal of forward thought anticipating the potential growth of the system.
Beyond these are many potential causes for data integration failure amongst model groups. The lack of a versioning strategy is one such cause. Defining an XML schema up front is ultimately fruitless as it will need to change over time (Wolk, 2006). If the schema, the shared store that supports it and every system that uses it has to move to a new version all at once, you cannot achieve data integration successfully. With services or integration of heterogeneous systems in general, a problem arises when there is a request for change of data representation which affects many documents and schemas (Klímek & Nečaský, 2010). Another cause for failure is a schema and store that require too much information. When people build a simple web service for point to point integration, they tend to think of the data their particular service needs.

2.4.10 Summary

This section presented an overview of the each of the different model groups which were common data models, data mapping, language models, schema mapping, meta-tagging models and layer models. The review and analysis of the groups of model approaches in this section has shown a diverse range of model approaches to data integration. It has examined the criticisms amongst the range of model approaches. Within each model group were similarities between the characteristics of the different approaches when contrasted against the classification scheme.

The common data models architecture included containing descriptions of data used by the organisation, a framework for data integration and a way to control data elements. The criticisms of this model are that it can be costly to develop and maintain, be developed too quickly thereby lacking in quality and often this model relies on a technology expert to develop or maintain it. They also lack support for new and emerging technologies due to be purpose specific for addressing current requirements.

Data mapping was identified as the backbone of the data integration process. However, the literature showed it has criticisms of being problematic within schema matching, schema mapping and model matching.
Language models are based on a specific query language as an approach to developing either data mappings or schema mappings. Criticisms included language models being often domain specific and with a specific purpose. They are not generic in nature.

Schema mapping models are one of the most common data integration model approaches. The literature showed there was a strong technical reliance in both implementing and maintaining schema mappings.

The data stream meta-tagging model approach involves writing meta-tags as part of achieving data integration. It was also showed to have a strong technical reliance in both implementation and maintenance.

Layer Models explained the high-level modeling approach to data integration based on a framework of layers, each a different step in the data integration process. This approach had criticisms of having strong technical reliance for maintaining and implementing the model and the problems associated with time by handling the way data sources change over time.

Other model types discussed that fall outside the characteristics discussed under the other groups of model approaches also had the same pattern of criticisms ranging from being domain specific through to complexity, reliance on a technical expert and high maintenance.

As a result of examining the different model groups and their characteristics against the dimensions discussed in the classification scheme, the importance of models in the data integration process is evident in aiding the approach to achieving successful data integration. However research showed that each of the model approach groups suffers a pattern of difficulties associated with data integration. These range from being domain specific, strong technical reliance for maintaining and implementing the model. There are problems associated with time by handling the way data sources change over time and lack of support for new and emerging technologies. The model approaches discussed are often implemented using web application technologies.
2.5 Technical approaches to data integration

2.5.1 Overview

Two approaches discussed in this chapter for integrating data sources for use in web applications are model approaches and technical approaches. A model approach lays down the framework for how an application approach is implemented. Technical approaches to data integration are methods that support or achieve data integration by means of utilising software tools. The software is built for a specific integration purpose or environment. Each application approach demonstrates a range of benefits.

Collectively, there is a multitude of existing and currently in development technical approaches for achieving data integration and each retains its validity and desirability in their respective areas. Technical approaches such as Standardisations, Brokers, Extensible Languages, Exposed APIs, Public APIs and Extract Transform Loads all boast strong support for data integration. However, many of these are intended to support integration within a specific domain or application area.

This section will focus on examining the literature around defining the types of technical approaches discussing the complexity, heterogeneity, the process and sources. The classification scheme introduced in section 2.3.3 is discussed to demonstrate the applicability of the different technical approaches for achieving data integration.

The dimensions introduced in section 2.3.3 will be used in an analysis to compare the characteristics of each of the technical approaches. In doing so, this lays down the groundwork for the following chapters and enables a discussion on the current approaches to data integration and the need for a more generic top level framework.

2.5.2 Standardisations

Technical approaches that have standardisation characteristics are focused on the streamlining and consistency of their data, to standardise the data from one data source
to another to exchange information across programs and systems. (Atre, 1998), (Goolsby & Levin, 2012). Data standardisation features two common terms; data cleansing and data scrubbing (Atre, 1998). Data cleansing is one of the most demanding parts of data warehousing and includes data scrubbing (Atre, 1998). These terms are described as interchangeable and both involve detecting and correcting (or removing) corrupt or inaccurate records from a database. Data must be cleaned. Otherwise, when raw data are aggregated, distilled, and summarized for use in the data warehouse, the inconsistencies and inaccuracies will distort and corrupt the summary views in ways that are difficult to trace or correct (Atre, 1998). The evolving application landscapes of companies will continue to become more and more complex as long as no standardization takes place (Schwinn & Schelp, 2005). Data cleansing services can transform and combine different data, remove inaccuracies, standardize common values, remove redundancy, parse values and cleanse corrupt data to create consistent, reliable information.

A positive relationship has been observed between software standardization and the predictability of the project scopes, schedules, and cost (Nidumolu, 1996). Overall, well-defined software processes provide control and predictability, increasing the likelihood of developing flexible systems (Sommerville, Sawyer & Viller, 1999). This has an immediate impact on data integration goals by improving the relationship between the workflow and data exchange of different systems can make the data integration process more seamless, streamlined and efficient (Goolsby & Levin, 2012). Consider the example in Table 2.11. The data is standardised to a consistent format from its source to its target. Software that achieves this examines the data for similarities and auto corrects the format so that there is consistency amongst the data elements. When the data reaches its target, it is then consistent when presented.

<table>
<thead>
<tr>
<th>WA</th>
<th>W.A.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Western Australia</td>
<td>West Australia</td>
</tr>
</tbody>
</table>

Standardised to:
In a case study example (Dorans, 2006), a company explains how the standardisation approach linked different applications together from different departments by integrating their data sources. Prior to standardizing on the software, the company used disparate applications for each department forcing a lot of data translating back and forth between different platforms, which made the process very time-consuming (Dorans, 2006). In another case example, the goal of standardisation is to provide data standardisation services for any incoming data, regardless of sources according to the Federal Information & News Dispatch (2000). In both cases, it is clear the objectives of standardization are about bridging sources together for streamlined flow and presentation of the data between them.

Many applications require specific platforms or versions. If the selected standard does not include the required platform, the value of standardization is eroded by the inability to provide the required service levels (Kahn, 2006). This leads to a lack of support for new and emerging technologies where new platforms are not considered by existing standardisations. In data integration, the standardisation approach can limit further integration capabilities if the applications change or evolve over time and the standardisation approach does not accommodate for such change. Labor productivity benefits are also suggested as a drawback. When an IT staff is trained on one toolset for a given function, there is a high switching cost (Kahn, 2006). The reliance upon IT and their involvement and in standardisation is high and time consuming.

In summary, standardisations are important in the data integration process as they enable data integrators to achieve data quality. But they lack support for new and emerging technologies. Given the architecture described above on how standardization technical approaches are structured, translation of data occurs during the work flow
between the source and the destination. The data is interpreted through the meta-tags before being delivered to the destination. The standardisation tool provides a description for how this translation occurs.

Standardisations contain programmable elements as the standardisation tool contains a set of definitions for defining the data and its attributes (Atre, 1998). These are written in a programming style language. Standardisations are centralised as they do encourage a single defining element to manage the integration process, via a centralised standardising platform (Goolsby & Levin, 2012).

Standardisations are not meta structured in nature. There are no meta-data elements within language Standardisations (Atre, 1998), descriptions are instead programmable in language and are query structured. Standardisations are not self-manageable, they are typically written by and controlled by an IT expert.

Table 2.12 compares this approach type with the dimensions of the classification scheme. It summarises these comparison discussions against the classification dimensions.
<table>
<thead>
<tr>
<th>Dimension</th>
<th>Dimension Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Translate</td>
<td>Yes</td>
</tr>
<tr>
<td>(2) Programmable</td>
<td>Yes</td>
</tr>
<tr>
<td>(3) Centralised</td>
<td>Yes</td>
</tr>
<tr>
<td>(4) Meta-data</td>
<td>No</td>
</tr>
<tr>
<td>(5) Self-manageable</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 2.12 Standardisation dimensions comparison

2.5.4 Brokers

Brokers use a software service approach to collect and integrate diverse data from autonomous data sources (Budgen, Rigby, Brereton & Turner, 2007) brokering the data between different systems (Clark, Krumm & Bieleski, 1992). Each system only interfaces to the Broker and does not exchange data directly with other systems. The Broker takes care of forwarding relevant data to all other systems. Delivering data through an information broker service offers the opportunity of integrating data from a range of autonomous sources while at the same time preserving any restrictions on access to or use of the information that those sources might impose (Budgen et al., 2007). An information broker has a distinct advantage over data-copying techniques, such as mining (Budgen et al., 2007). They act as a negotiator and a medium between the data source and its destination. The broker is able to collect the data from the source and determine how and where it goes towards its destination, acting as a negotiator between the source and the destination. Brokers are an important technical approach to data integration as they have a distinct advantage over data-copying techniques, such as mining and enforce a data as a service approach (Budgen et al., 2007).

To illustrate an example in more detail, Figure 2.11 presents a UML component diagram of a broker workflow. The broker acts as a negotiator between the data sources and the user interface which is accessing the data. The services process the queries and responses from the data sources and map a query into a form that matches the data structure and semantics of a particular data source. The broker collects the data via the web services and interprets and determines where the data fits with the user access
level. It represents each data source and is the access medium for that data source. The benefits of brokers stem from the relationship between the broker and the web services resulting in the information the broker can use without needing any detailed knowledge of how the data is organized or managed (Budgen et al., 2007).

![UML component diagram of broker workflow](image)

Figure 2.11 UML component diagram of broker work flow

The broker ensures the autonomy of each data source from its processes negotiating with web services (Budgen et al., 2007). This relationship is broken up into five layers which are as described in Table 2.13.
<table>
<thead>
<tr>
<th>Description</th>
<th>Encompasses characteristics such as functionality, interfaces, nonfunctional characteristics and constraints, as well as the parameters the service provider sets for service provision (Budgen et al., 2007).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discovery</td>
<td>Supports the location of client services, typically by identifying a list of candidate services and providers (Budgen et al., 2007).</td>
</tr>
<tr>
<td>Negotiation</td>
<td>Dynamic interaction between the client and one or more providers to agree on the terms and conditions for supplying a service. Negotiation usually involves matters of contract and pricing (Budgen et al., 2007).</td>
</tr>
<tr>
<td>Delivery</td>
<td>Involves the invocation, provision, and suspension of the service itself (Budgen et al., 2007).</td>
</tr>
<tr>
<td>Composition</td>
<td>The process by which an end user or another service creates services that is not atomic from a set of lower level services. It either uses history to select these or identifies suitable ones on demand (Budgen et al., 2007).</td>
</tr>
</tbody>
</table>

**Table 2.13 Broker characteristics by 5 layers**

Figure 2.12 illustrates where these layers site within the broker and web service relationship. Each layer describes the process of how the data is queried and flows between web services, through the broker to the end user.
In summary, brokers are an effective approach in the data integration process. They enable data integrators to achieve integration with multiple sources preserving different rules and protocols (Budgen et al., 2007). They do this by acting as a medium and negotiator between the sources and the destination. Each system can interface to the Broker in whatever format is most easily implemented. This is important for legacy systems (Clark, Krumm & Bieleski, 1992). From the architecture and characteristics described above on how brokers are structured, translation of data occurs during the workflow between the broker and the web services. The data is interpreted through the broker before being delivered to the destination.

Brokers contain programmable elements as the broker tool contains a set of definitions for defining how data is interpreted. These are written in a programming style language. Brokers are centralised as they do encourage a single defining element to manage the integration process, via a centralised broker.

Brokers are not meta structured in nature (Budgen et al., 2007). There are no meta-data elements within brokers, descriptions are instead programmable in language and are query structured. Brokers are not self-manageable, they are typically written by and controlled by an IT expert (Budgen et al., 2007).
Table 2.14 summarises these comparison discussions against the classification dimensions. It compares this approach type with the dimensions of the classification scheme.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Dimension Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Translate</td>
<td>Yes</td>
</tr>
<tr>
<td>(2) Programmable</td>
<td>Yes</td>
</tr>
<tr>
<td>(3) Centralised</td>
<td>Yes</td>
</tr>
<tr>
<td>(4) Meta-data</td>
<td>No</td>
</tr>
<tr>
<td>(5) Self-manageable</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 2.14 Broker dimensions comparison

2.5.5 Extensible languages

In its simplest form, extensible languages are the need for two systems to be able to communicate when they have a common vocabulary but not complete understanding of all the features they each use (Akpotsui, 1997). Extensible programming languages enable programmers to introduce customary language features as language extensions of the base language (Erdweg & Rieger, 2013). The most commonly used extensible language is Extensible Markup Language. It is impossible to overstate the importance of XML as a data organization tool (Berman, 2005). Extensible Markup Language (XML) is a data organization tool (Berman, 2005). XML is a method for marking up files so that every piece of data is surrounded by bracketed text that describes the piece of data (Berman, 2004). “HTML for displaying data, XML for describing it” (Geroimenko, 2003). XML is a language to go beyond the static limitations of HTML describing not only the style but also the content of a web document. XML marks up the content not only in a meaningful way that can be understood by human beings but also by computers (Geroimenko, 2003).

The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries and involves publishing in languages specifically designed for data such as XML according to the
World Wide Web Consortium (2011). Semantic Web is a tool that proposes to allow integration of disparate datasets according to Management Association, Information Resources (2014). It involves publishing in languages specifically designed for data being XML Syntax, RDF Data Interchange, URI Identifiers and Ontologies World Wide Web Consortium (2011). Figure 2.13 illustrates the architecture and relationship between each of the Semantic web elements.

![Figure 2.13: Semantic web elements](image)

XML Syntax involves the markup of the data in XML or a similar format. Markup is the process of annotating data with meta-data which in turn results in conveying any message as XML. XML documents fall into 1 of 2 different types: structural or data-centric (Akpotsui, 1997). It involves arranging the data elements in Resource Description Framework (RDF) statements (Information Resources Management Association, 2014). RDF is simply a data model, a way to express complex data in simple, straightforward sentences (Information Resources Management Association, 2014) to take a complex dataset and break it down into its simplest form. To illustrate an example of how this works in more detail, consider the following fields:

Date: 24/04/2016 Customer Jeremy Nunn

XML labels describe these fields

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns">
  <rdf:Description rdf:about="Customer field">
    <date>24042016</date>
    <customer>Jeremy Nunn</customer>
  </rdf:Description>
</rdf:RDF>
```
The Semantic Web involves identifying each data element with a URI and making each data element available via that URI (Information Resources Management Association, 2014). RDF is a standard model for data interchange on the web (Information Resources Management Association, 2014). It has features that facilitate data merging even if the underlying schemas differ. It supports the evolution of schemas over time without requiring all the data consumers to be changed (Information Resources Management Association, 2014). RDF extends the linking structure of the web to use URIs as well as the two ends of the link XML according to the World Wide Web Consortium (2013). URI’s are like URLs but they are actual locations and not just pointers to locations (Information Resources Management Association, 2014).

With the Semantic Web, many existing ontologies can be reused to label data elements (W3C, 2009). Ontologies are structured vocabularies that RDF statements refer to provide meaning to their statements (Information Resources Management Association, 2014) adding more vocabulary for describing the properties of the data element such as through a Web Ontology Language OWL (W3C, 2009).

Systems that support language extensions are either agnostic to the base language or only support a single base language (Erdweg & Rieger, 2013). This indicates extensible languages are a domain specific approach, limited to supporting the base programming language a web application was written in.

In summary, extensible languages are common data formats on the World Wide Web (W3C, 2009) and also enable the semantic web as discussed above. When attempting a data integration process, extensible languages are a method to assist data integrators to achieve integration with multiple sources interchanging common data formats and the benefits of having data that describes the data being integrated with. From the architecture and characteristics described above on how are extensible languages are structured, translation of data occurs during the work flow between the elements of the semantic web. The data is interpreted through these elements before being delivered to the destination.
Extensible languages contain programmable elements (Erdweg & Rieger, 2013) as the extensible language contains a set of definitions and tags for defining and describing the data being interpreted. These are written in a programming style language.

Extensible languages are not centralised as they do not have a single platform for managing the integration process. Publishers use common ontologies which provided common meaning to data, but that is not the same thing as management. Extensible languages can be meta structured in nature (Erdweg & Rieger, 2013). XML for example is made up of meta-data elements which describe the data within this element. Extensible languages are not easily managed, they are typically written by and controlled by an IT expert. Deep knowledge is required on how to write code expressions and maintain an extensible language. A data integrator who is a blogger would need to learn this knowledge or rely on an IT expert to implement an integration process that relies on extensible languages as an integral part of it. Automated tools that can be installed into a web application still require technical knowledge to implement, configure and integrate with the web application.

Table 2.15 summarises these comparison discussions against the classification dimensions. It compares this approach type with the dimensions of the classification scheme.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Dimension Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Translate</td>
<td>Yes</td>
</tr>
<tr>
<td>(2) Programmable</td>
<td>Yes</td>
</tr>
<tr>
<td>(3) Centralised</td>
<td>No</td>
</tr>
<tr>
<td>(4) Meta-data</td>
<td>Yes</td>
</tr>
<tr>
<td>(5) Self-manageable</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 2.15 Extensible languages dimensions comparison

2.5.6 Exposed APIs
Application Programming Interfaces (APIs) are a description of the way one piece of software asks another program to perform a service (Orenstein, 2000). It is a specification created to be used as an interface consisting of software libraries, modules, data structures, object classes and variables which allows different software components to communicate and interact with each other (Orenstein, 2000).

Classes describe the behavioral and structural fields of a data set. Table 2.16 illustrates what a class would look like.

<table>
<thead>
<tr>
<th>Company</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company name</td>
</tr>
<tr>
<td>Company address</td>
</tr>
<tr>
<td>ABN</td>
</tr>
<tr>
<td>Active</td>
</tr>
<tr>
<td>Status</td>
</tr>
<tr>
<td>Registered</td>
</tr>
</tbody>
</table>

Table 2.16: A sample class

An API includes a description of a set of class definitions. An exposed API defines the methods to interact with the objects defined in class definitions (Orenstein, 2000). Figure 2.14 illustrates a UML Composite structure diagram of an API. All the methods of the API are exposed by the class definitions thereby making this functionality available to be integrated with via other software components or data sources.
In Figure 2.14, the work flow starts with a software system and its API describing how to integrate with it. This API consists of class definitions exposing functionality to integrate with. The classes allowing for other software applications to interact with the data structure and definitions detailed in the classes and achieve integration.

In summary, Exposed APIs are a common technical approach to integrating a web application with external extensions to be used within the application. From the architecture and characteristics described above on how are Exposed APIs are structured, translation of data occurs during the work flow between the API, the web application and its source extensions. The data is interpreted through the API before being delivered to the web application. Exposed APIs contain programmable elements and these are written in a programming style language.
Exposed APIs are not centralised as they do not encourage a single defining element to manage the integration process. Exposed APIs are meta structured in nature. The elements within the API are made up of meta-data elements which describe the data within each function, class or element. Exposed APIs are not self-manageable, they are typically written by and controlled by an IT expert.

Table 2.17 compares this approach type with the dimensions of the classification scheme.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Dimension Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Translate</td>
<td>Yes</td>
</tr>
<tr>
<td>(2) Programmable</td>
<td>Yes</td>
</tr>
<tr>
<td>(3) Centralised</td>
<td>No</td>
</tr>
<tr>
<td>(4) Meta-data</td>
<td>Yes</td>
</tr>
<tr>
<td>(5) Self-manageable</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 2.17 Exposed APIs dimensions comparison

### 2.5.7 Public APIs

A Public API in simple terms is exposing key pieces of your application for the consumption of other applications (Musser, 2010). Beyond the benefits of exposed APIs, Public APIs expose key pieces of the application for integration by other applications. This is an important goal of data integration as the integration approach may only seek to integrate part of the application with external applications and sources.

Using open standards, Public APIs enable developers to using your API to further expand and develop the application beyond the scope of that the creators of the API had originally intended (Musser, 2010). Figure 2.15 illustrates a UML Component Diagram of the workflow for how an application is accessible via Public APIs to external applications to integrate with. The web application makes one of its modules available via a public API to external applications. The application has two modules, one of which it is sharing
via a Public API which enables external applications to integrate with this module from the application (Musser, 2010).

Figure 2.15: Public APIs workflow

The framework behind public APIs consists of REST v/s Web Services (SOAP), Response Data Format (XML, JSON), Service Contract Description, API Authentication, Service Versioning, Rate Limits, Documentation and Helper Libraries features (Musser J 2010).

Common limitations of Public APIs are that they are application specific and require a very heavy technical implementation and ongoing technical maintenance commitment (Musser, 2010).
In summary, Public APIs are a common technical approach to integrating a web application with external applications by enabling one of its modules to be available via a public API to external applications. From the architecture and characteristics described above on how are Public APIs are structured, translation of data occurs during the workflow between the API, the web application and the API and its source extensions. The data is interpreted through the API before being delivered to the web application. Public APIs contain programmable elements and these are written in a programming style language.

Public APIs are not centralised as they do not encourage a single defining element to manage the integration process. Public APIs are meta structured in nature. The elements within the API are made up of meta-data elements which describe the data within each function, class or element. Public APIs are not self-manageable, they are typically written by and controlled by an IT expert.

Table 2.18 compares this approach type with the dimensions of the classification scheme.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Dimension Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Translate</td>
<td>Yes</td>
</tr>
<tr>
<td>(2) Programmable</td>
<td>Yes</td>
</tr>
<tr>
<td>(3) Centralised</td>
<td>No</td>
</tr>
<tr>
<td>(4) Meta-data</td>
<td>Yes</td>
</tr>
<tr>
<td>(5) Self-manageable</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 2.18 Public APIs dimensions comparison

2.5.8 Extract transform loads

Extract transform loads (ETL) are a process in database usage for extracting data from outside sources and transforming it and loading it into the target source such as a data warehouse and presented to the user through a data mart (Kimball, 2008). This process is effectively an exporting and importing process involving some transformation of the
data along the way and various integrity checks. Figure 2.16 illustrates a UML Component Diagram showing how the data flows between the ETL and the data warehouse. The ETL extracts the data from the data sources, transforms it and presents it at the data warehouse. From the data warehouse, the data mart presents it to the user.

![UML Component Diagram](image)

Figure 2.16: UML component diagram of extract transform loads work flow

In general ETL are used to load data warehouses and are the area where the business intelligence information is housed according to the ETL Tools Poll Results (2012). ETL tools extract data from underlying data sources via both native DBMS gateways (e.g. from relational vendors, ISAM, etc.) and via standard interfaces like ODBC; they then load the data into a warehouse. Typically, an ETL tool also provides a facility to specify data transformations, which can be applied as the data is being extracted from the data sources and loaded into the warehouse (Hellerstein, Stonebraker & Caccia, 1999).
A main problem with ETL tools is that they were designed to solve a niche problem for warehouse loading and are not useful for general data transformation (Hellerstein, Stonebraker & Caccia, 1999). Another limitation of ETL is that it cannot support real time transfers (Kimball, 2008). This is a major shortcoming for support data integration processes for use in web applications as they typically rely on real time data. These two areas highlight how ETL are another approach that is domain specific for solving a specific integration task or problem and has difficulties with other approaches and that it is not a generic solution that can be applied to all integrating scenarios.

Table 2.19 compares this approach type with the dimensions of the classification scheme.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Dimension Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Translate</td>
<td>Yes</td>
</tr>
<tr>
<td>(2) Programmable</td>
<td>Yes</td>
</tr>
<tr>
<td>(3) Centralised</td>
<td>No</td>
</tr>
<tr>
<td>(4) Meta-data</td>
<td>Yes</td>
</tr>
<tr>
<td>(5) Self-manageable</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 2.19 Extract transform load dimensions comparison

2.5.9 Analysis

The importance of technical approaches in the data integration process is evident in aiding the approach to achieving successful data integration outcome. Each category of approaches has its application and benefits but also has a series of difficulties associated with data integration.

Standardisations are focused on the streamlining and consistency of data, standardising data from one data source to another. It is an integral component in a data integration process that relies on a work flow of data being synched and standardised between two data sources. The shortcoming being that some data integration processes cannot be
standardised due to constantly changing or unpredictable data types. Standardisation relies on a form of prediction to determine what the data looks like to standardise it.

Brokers are reliant on a service based approach to negotiate or act as a medium between the data source and its destination. In not all instances can a broker be implemented as part of a data integration process. It relies on predicting what the data looks like to negotiate with the destination. Brokers are not self-manageable and require high-level technical implementation to implement and maintain.

Extensible languages rely on a consistency in language between the destination and the source. In the event the destination and the source communicate differently to each other, this approach falls short. Extensible languages are also reliant on predetermining what the data is going to look like to interpret it. For constantly changing data, a high-level of technical implementation and maintenance is required meaning this approach is not self-manageable.

Exposed and Public APIs are both reliant on predetermined explanations of what the data is going to look like. They are both also heavily reliant on high-level technical implementation and maintenance.

Extract Transform Loads are tools to extract data from underlying data sources. A main problem with ETL tools is that they were designed to solve a niche problem such as warehouse loading and are not useful for general data transformation. Furthermore, Extract Transform Loads face a major limitation when integrating with real time data sources which is primarily what web applications rely on.

2.5.10 Summary

The review and analysis of the groups of technical approaches in this chapter have showed a diverse range of technology groups for use in a data integration process. There were similarities in the criticisms of each application approach group. When contrasted against the classification scheme and the different dimensions, there were often only a few distinguishing characteristics between the approaches.
The design space of technical approaches is described in terms of their dimensions. These classifications were used to further test the dimensions of the technical approaches described in this section.

The different groups of technical approaches discussed included standardisations, brokers, extensible languages, exposed APIs, public APIs and extract transform loads. Each approach architecture and benefits were discussed in supporting data nitration data sources for web applications. The characteristics of each approach were contrasted against the dimensions of the classification scheme. The analysis showed the importance of these technical approaches in the data integration process and their comparison against each of the dimensions for the classification of application types.

Technical approaches were shown to be problematic. The autonomy of the data integration process can lead to uncontrolled redundancy and support for the design process of data integration solutions (Schwinn & Schelp, 2005). Many of these are intended to support integration within a specific domain. They are each field specific and designed to solve data integration problems for a specific technical environment. Many approaches also do not support real time integration. They are also heavily reliant on high-level technical implementation and maintenance. There is also a lack of support for new and emerging technologies.
Table 2.20 illustrates the different technical approaches in comparison to each other against the criteria discussed:

<table>
<thead>
<tr>
<th>Model Type</th>
<th>Translation</th>
<th>Programmable or Fixed</th>
<th>Centralised or DeCentralised</th>
<th>Meta-data</th>
<th>Self Manageable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standardisation</td>
<td>Yes</td>
<td>Fixed</td>
<td>Centralised</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Brokers</td>
<td>Yes</td>
<td>Fixed</td>
<td>Centralised</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Extensible Languages</td>
<td>Yes</td>
<td>Programmable</td>
<td>DeCentralised</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Exposed APIs</td>
<td>Yes</td>
<td>Programmable</td>
<td>DeCentralised</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Public APIs</td>
<td>Yes</td>
<td>Programmable</td>
<td>DeCentralised</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Extract Transform Load</td>
<td>No</td>
<td>Fixed</td>
<td>DeCentralised</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 2.20 Application comparison against classification scheme

At a high-level, there is a single overseeing framework that is still missing that can be applied to these areas that could address the unique challenges identified within.

2.6 Management process-based approaches

Model and technical approaches have been identified as having a series of difficulties associated with data integration as discussed in sections 2.4 and 2.5. There are approaches that exist for addressing these difficulties such as management process-based approaches which could be used to improve integrating data sources for use in web applications. In the context of the objectives of this thesis, the management process-based approaches discussed will be various software development processes, methodologies and frameworks.
This section discusses these various management process-based approaches that could be used to support the integration process. It examines the viability of each group of management process-based approaches contrasting the characteristics of each against a classification scheme introduced in this chapter in section 2.3.3. The processes consist of extending existing frameworks, adapting an IT management framework or altering software development frameworks and toolkits. It examines what each management process is, what it would involve adapting one for supporting a data integrating in their integration process.

### 2.6.1 Extending existing frameworks

A framework is defined as a class hierarchy plus a model of interaction among the objects instantiated from the framework (Lewis, 1995). In addition, a framework reverses the traditional idea of components reuse. Instead of a programmer writing a main program which calls reusable procedures, they instantiate objects from the frameworks class hierarchy and then provide methods for the framework to call. Thus, a framework is a generic application program which a programmer tailors by providing highly specialised routines which are called by the framework (Lewis, 1995). This means that frameworks are an incomplete workflow that needs to be adapted to meet the specific requirements of the application. A framework represents a generic solution. It is a meta-solution encompassing a set of possible solutions, rather than any one solution, within a particular problem domain. A framework reflects many solutions in the domain at once, without necessarily solving any one particular problem (Roberts & Johnson, 1998). It acts as a universal, reusable software platform to develop applications.

Extending existing frameworks is an approach adding new features to an existing framework in a way that preserves existing customisations, configurations and other extensions of that framework (Tourwe, 2002). This might include further development of the existing frameworks features and workflow or adding entirely new workflow to it. In the context of this thesis, extending an existing framework would involve creating additional features to enable it to be applied as a solution for a data integrator to use when integrating data sources for use in web applications. This means taking an existing framework such as the Generic Framework for Information Management (van Bon,
Verheijen, 2006) and manipulating it with the addition of an extension for the specific purposes of integrating data sources for use in web applications.

With the example of the Generic Framework for Information Management (van Bon & Verheijen, 2006), this is a model for the interrelating of different components of information management. It is geared towards analysis of organisational and responsibility issues. It is used to support strategic decision making in three different ways. The first is descriptive orientation where the framework maps the information management domain (van Bon & Verheijen, 2006). The second is specification design where it is used for re-organising the information management organisation (van Bon & Verheijen, 2006). The third being prescriptive normative used as a diagnostic instrument to map out gaps in the organisations information management highlighting missing interrelationships between the various components of the framework (van Bon & Verheijen, 2006).

Extending this framework for the purpose of integrating data sources for use in web applications would involve developing new features across the three different components and changing the framework scope. The frameworks first component would be extended for mapping out the integration activity. Developing new features would include adding to the specification component so that it is focused on the re-organisation of data sources rather than the re-organisation of information management. The diagnostic instrument would be extended for mapping out gaps in the integration activity. Extending the frameworks scope would re-model it from interrelating different components of information management to interrelating different components of data sources for integrating with web applications. While the components of this framework would remain the same in structure, there would need to be extensions to them to achieve the new aims. The challenge here is that the framework is not a technical one. It requires significant remodeling to be suitable for use with web applications and it involves basing the solution to integrating data sources with web applications on a framework not designed specifically to address this challenge.

Microworkflow is a framework for building workflow applications (Manolescu, 2000). In this example, extending the Microworkflow framework would involve programming elements. Microworkflow is a technical framework and creating extensions to it involves
significant coding work. Microworkflow would remain as the base framework and the resulting extension of the framework with the new functionality would become its own new framework.

Figure 2.17 illustrates extending the Microworkflow framework. In this example the data integrator has a decision making stage to determine which framework to choose for their integration process. Each one has its own characteristics and in this example microworkflow is the chosen framework. The Microworkflow is extended with new elements for the specific integration process the data integrator intends to go through which involves programming these new elements. The resulting new framework is separate to the original base framework once these extensions are complete.

Figure 2.17: New framework from extensions

Criticisms of extending frameworks are that they are often designed to be a one size fits all approach (Roberts & Johnson, 1998). Framework development demands a continuous process of improvement (Roberts & Johnson, 1998) and programs developed using multiple frameworks are prone to have interaction problems between them (Bosch, Molin, Mattson, Bengtsson & Fayad, 1999).

The classification scheme introduced in section 2.3.3 varied between frameworks but from a high-level view of applying it to the area of extending frameworks, various characteristics are notable. Translation still occurs with the data flow between sources. The framework is programmable but it isn’t centralised since it is dependent on someone developing and implementing the framework. Meta-data exists as extending a framework does provide meta-descriptions of the data workflow. The extension of the framework isn’t self-manageable as it may require a technical writer to develop the extension or
someone with extremely high knowledge of the original framework workings to successfully expand it.

Table 2.21 compares extending frameworks with the dimensions of the classification scheme.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Dimension Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Translate</td>
<td>Yes</td>
</tr>
<tr>
<td>(2) Programmable</td>
<td>Yes</td>
</tr>
<tr>
<td>(3) Centralised</td>
<td>No</td>
</tr>
<tr>
<td>(4) Meta-data</td>
<td>Yes</td>
</tr>
<tr>
<td>(5) Self-manageable</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 2.21 Extending existing frameworks

2.6.2 Adapting an IT management framework

IT management frameworks are a wide encompassing topic but in the context of this thesis, they are a management framework approach that can be used for integrating data sources for use in web applications. Applying this approach to integrating data sources for use in web applications involves modifying taking an existing IT management framework and modifying it for the specific purposes of a data integration process. This could involve additions to the IT management framework to make it work for integration purposes. Contrast to the extension of a framework where the original framework remains as a base framework and the extensions form a new framework, adapting a framework involves working with just the base framework.

Several frameworks exist to help organizations in IT management implementation. However, organisations still prefer to design their own (Broussard & Tero, 2007). The body of literature suggests most frameworks state that there is no single best IT management or organisational structure or governance arrangement because IT needs to respond to unique environments (Pereira & da Silva, 2012).
In Figure 2.18 the base framework is adapted by taking the features from the framework applicable to the data integration task and implementing the framework in the process. The result being the framework successfully applied to the integration activity.

![Figure 2.18: Adapting a framework to an integration activity](image)

Six Sigma is a quality and process improvement focused IT management framework. It is an adaptable methodology focused on measurement based improvements that are used for delivering quality IT outcomes according to IT Governance (2014). The main aim is to reduce variation in processes through a structure by which organisations can constantly improve IT processes and eliminate defects, waste and expense (IT Governance, 2014). In this example, adapting it for the purposes of integrating data sources for use with web applications would mean implementing measurement based improvements to the integration process to ensure a high quality outcome. This might involve identifying what difficulties exist with the integration activity and establishing measurable improvements to the integration process. In the identification phase, any defects are removed to improve the quality of the integration activity. It is a process structured approach but heavily reliant on a highly experienced Six Sigma practitioner. Factors that can influence each IT management frameworks implementation includes being seen as complex (Pereira & da Silva, 2010), too general (Morimoto, 2009), and overlap each other (Pereira & da Silva, 2010).

IT management frameworks on their own would not act as an overarching mechanism to support a data integrator in their data integration activity. However, they would contain
elements that do support the data integrator in their integration activities. But it is the factors described above that indicate a similar group of difficulties observed with other approaches discussed in this chapter. IT management frameworks are criticised as being too complex, too general and even overlapping each other. When contrasting the characteristics of IT management frameworks to the classification scheme proposed in this chapter, this approach does see translation occur during a data integration process, it contains meta-data elements and it does contain programmable elements. However it is not a centralised approach or self-manageable because of its complexity, reliance on highly skilled individuals and in some cases generalised nature.

Table 2.22 compares the characteristics of IT management frameworks discussed against the classification scheme proposed in this chapter. It compares this approach type with the dimensions of the classification scheme.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Dimension Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Translate</td>
<td>Yes</td>
</tr>
<tr>
<td>(2) Programmable</td>
<td>Yes</td>
</tr>
<tr>
<td>(3) Centralised</td>
<td>No</td>
</tr>
<tr>
<td>(4) Meta-data</td>
<td>Yes</td>
</tr>
<tr>
<td>(5) Self-manageable</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 2.22 Adapting an IT management framework

2.6.3 Altering software development frameworks

A software development framework is a software life-cycle process in the management process category. This category consists of a set of processes that contain procedures that can be used by managers for any type of software development project (de Amescua, Garcia, Velasco & Martinez, 2004). Adapting a software project management framework is an approach aimed at modifying an existing software development framework such as PRINCE2 to apply it as an approach for integrating data sources for use in web applications. This means taking an existing framework and adapting it for the specific purposes of integrating data sources for use in web applications. This would
involve additions to the framework to make it work for integration purposes. Contrast to
the extension of a framework where the original framework remains as a base
framework and the extensions form a new framework, adapting a framework involves
working with just the base framework.

PRINCE2 is a software management framework. It is a methodology that encompasses
the management, control and organisation of a data integration project (Richards, 2007).
It is a process driven and structured project management approach describing the
procedures to coordinate the people and activities of a project, how to design and
coordinate the project (Richards, 2007). Criticisms include PRINCE2 being considered
inappropriate for small projects due to the strong reliance on team member project
planning according to the Office of Government Commerce (2002) or where project
requirements are expected to change rapidly (Richards, 2007). There are also high
maintenance criticisms for PRINCE2 projects (Richards, 2007). PRINCE2
documentation implemented in an overly bureaucratic way described as a common
mistake (Richards, 2007). PRINCE2 also requires a high-level of training for team
members in the project (Office of Government Commerce, 2002). Data integration
projects deal with unpredictable data and as the project evolves, the requirements will
change rapidly. Adapting an existing framework like PRINCE2 for improving integration
of data sources for use in web applications wouldn’t be the most appropriate approach
for use with web applications. The criticisms of this approach highlight the limitations in
meeting these goals too.

The nature of the data integration process involves quick execution and handling of
unpredictable data sets. This is where software development frameworks such as Scrum
(Agile methodology wrapping extreme programming) (Fowler & Highsmith, 2001),
Unified Process (Iterative and incremental software development process framework) or
Extreme Programming (Jeffries, Anderson & Hendrickson, 2001) have been common
methodology groups adopted in a data integration process because of their flexible
nature being an agile methodology.

The Agile methodology is an approach aimed at fast paced implementation delivering
results earlier in the project and accommodating changing requirements (Fowler &
Highsmith, 2001). It is based on principles of frequent software releases to the customer
or user, welcoming changing requirements in the project and team collaboration (Fowler & Highsmith, 2001). It is a group of software development methods based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams.

Scrum is an agile software development method that is used for software development projects and business integration projects. Scrum is a type of iterative, incremental Agile software development framework for software development project management (Hong, Yoo & Cha, 2010). There are a growing number of companies that profess to be using agile methodologies on business integration projects, in particular Scrum and Extreme Programming (Hong et al., 2010).

A survey conducted in Oct. 2008 by the Cranky Product Manager Polls & Surveys (2008) reported that more than 60% of the software industry utilized the waterfall methodology until 2006. In 2008 the trend shifted to agile methodologies (Hong et al., 2010). With the waterfall methodology, the success of a project lies solely with the project manager compared to an agile methodology which focuses on team work and cooperation to deal with any changes in project requirements (Hong et al., 2010).

Some criticisms of Agile methodologies are that they result in over responding to a change in requirements which has been cited as the source of many software disasters (Boehm, 2002). The literature also suggests that complaints around the data being the problem is actually a result of not understanding that data management is the most important aspect of the business integration process and not just data delivery (Moss, 2009). It also suggests the more complex the system architecture and the software are, the more thinking behind the architecture that has to be done before coding can begin (Moss, 2009).

Data integrators waiting for the data to be ready in the data warehouse before there are developed selected integration features can often be accomplished within days or weeks (Moss, 2009). Developing the front end integration application is only one of 16 data warehouse development steps (Moss, 2009). The other 16 development steps have activities and tasks that require an enterprise perspective, such as data standardization, data integration, enterprise data modeling, business rules ratification, coordinated data
staging, common meta data, collectively architected (designed) and databases (Moss, 2009). Waiting for the data to be ready in the data warehouse means they will develop their own solo integration solutions by going directly against the operational source systems. But in this example Extreme Programming would be a preferred methodology because of the type of front end effort required for the integration process. In Extreme Programming the emphasis is on programming, building the system in small releases so that the customer benefit is maximized and achieving the best possible feedback on how it is progressing (Jeffries et al., 2001). Applying this to an integration process, the integration with data sources would be done in small stages to test the results on how it is progressing.

Only five development steps have a purely narrow project focus: project planning, requirements definition, prototyping, integration application development, and implementation (Moss, 2009). For some projects, the steps of prototyping and integration application development may also require an enterprise perspective because of some shared functionality or common reporting. The literature suggests neither Scrum nor extreme programming take any of these additional data warehouse specific complexities and dependencies into account (Moss, 2009).

Figure 2.19 illustrates the workflow a data integrator would go through when extending some of the existing software development frameworks discussed.

![Figure 2.19 Extending existing software development frameworks](image)

In Figure 2.19 the data integrator has a decision making stage to determine which software development framework to choose for their integration process. Each one has
its own characteristics. PRINCE2 in this example relies on strong planning and heavy team involvement versus Scrum which is an agile approach but has criticisms where the implementation can result in over responding to change. The framework alterations adapt the chosen framework for the specific integration process involving modifying the framework processes to make it work for the specific integration purpose. This could involve changing the order of the framework steps, applying an emphasis on some steps over others or modifications to just one aspect of the framework. For example when using PRINCE2 as an approach, the data integrator may make most of their modifications around the planning stage and follow the team involvement process as is.

The current software development frameworks discussed would not be appropriate as the sole mechanism for supporting a data integrator in their data integration activities. The characteristics of current software development frameworks discussed in this section show that these frameworks, individually in their current form, are not appropriate as an approach for supporting all integrating scenarios for integrating data sources for use in web applications. They are found to be domain or purpose specific. There is a reliance on an IT expert and the more complex the system architecture and the software are, the more thinking behind the architecture that has to be done before coding can begin. The literature also suggested that there can be over responding to a change in requirements. Software development frameworks are however extensible so that they can be altered for the specific purpose of integrating data sources for use in web applications. They can be built on so that they can be applied for this purpose and used to support a data integrator in their integration process. Table 2.23 compares this approach type with the dimensions of the classification scheme.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Dimension Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Translate</td>
<td>Yes</td>
</tr>
<tr>
<td>(2) Programmable</td>
<td>Yes</td>
</tr>
<tr>
<td>(3) Centralised</td>
<td>No</td>
</tr>
<tr>
<td>(4) Meta-data</td>
<td>Yes</td>
</tr>
<tr>
<td>(5) Self-manageable</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 2.23 Adapting a software development framework
2.6.4 Altering toolkits

Toolkits are typically software interfaces designed to enable access to data in real-time according to DNB (2014), (Chen & Wu, 2005), and the International Journal of Micrographics & Optical Technology (2010). Applying this approach to integrating data sources for use in web applications involves modifying an existing development toolkit to suit the goals of integrating data sources for use in web applications. This means taking an existing development toolkit and enhancing it or modifying it for a specific data integration process.

An example toolkit is the DNB The Data Integration Toolkit™ which is a software tool designed to manage multiple information sources to obtain consistent data from around the world accessing over 180 million business records (DNB, 2014). It's based on an XML-based Delivery Technology and Pre-integrated Solutions compatible with leading ERP, CRM and e-Commerce providers including Oracle and SAP (DNB, 2014).

DartGrid which is based on the semantic web is another toolkit example. It uses RDF-based ontology to integrate heterogeneous relational databases (Chen & Wu, 2005). By uniformly defined domain semantics, data resources are semantically registered to a semantic registration service and seamlessly integrated together to provide unified semantic query and semantic search service (Chen & Wu, 2005). Furthermore, a set of semantic tools are developed to raise the level of interaction with the system to a semantic level. A search-engine-like interface is offered to enable quickly search and semantically navigate data from one database to another database (Chen & Wu, 2005).

FlexiCapture Engine is a software development toolkit for integrating data capture technology into third party applications according to the International Journal of Micrographics & Optical Technology (2010). It consists of a comprehensive set of tools for rapidly developing automated processing of various types of forms and documents and extraction of key data for input into backend systems (International Journal of Micrographics & Optical Technology, 2010).

In each example the toolkit is not web application focused and each is a technology specific approach. Each of the toolkits discussed are designed to solve a specific
technical challenge and service a target technical environment. They are not designed to be a generic approach that could be used across potentially vastly different technical environments. Instead, each toolkit is single purpose specific. Altering one for the purposes of integrating data sources for use in web applications would be limited to the specific technologies and technical environment the toolkit was designed for.

In Figure 2.20, the data integrator has a decision making process to determine which toolkit to use for their integration process. In this example DataGrid is the chosen toolkit and the alterations to it for the specific integration purpose involve programming new elements to the toolkit. These alterations extend the original framework resulting in a new version of the toolkit separate from the original base toolkit.

Figure 2.20 illustrates an example extending the DataGrid toolkit

When contrasting the characteristics of altering toolkits to the classification scheme proposed in this chapter, this approach does see translation occur during a data integration process, it contains meta-data elements and it does contain programmable elements. Due to its complexity, it is not a centralised approach or self-manageable.

Table 2.24 compares this approach type with the dimensions of the classification scheme.
### Table 2.24 Altering toolkits

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Dimension Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Translate</td>
<td>Yes</td>
</tr>
<tr>
<td>(2) Programmable</td>
<td>Yes</td>
</tr>
<tr>
<td>(3) Centralised</td>
<td>No</td>
</tr>
<tr>
<td>(4) Meta-data</td>
<td>Yes</td>
</tr>
<tr>
<td>(5) Self-manageable</td>
<td>No</td>
</tr>
</tbody>
</table>

#### 2.6.5 Summary

This section reviewed the groups of management approaches that can be applied as an approach for supporting integration of data sources for use in web applications. There were similarities in the criticisms of each management approach group. When contrasted against the classification scheme and the different dimensions, there were often only a few distinguishing characteristics between the management approaches.

The design space of data integration management approaches is described in terms of their dimensions. These classifications were used to further test the dimensions of the management approaches contrasting the characteristics of each against a classification scheme introduced in this chapter in section 2.3.3.

The management processes consist of extending existing frameworks, adapting an IT management framework or altering software development frameworks and toolkits. Extending existing frameworks might include further development of the existing frameworks features and workflow or adding entirely new workflow to it. A restriction with extending a framework is that they are often designed to be a one size fits all approach.

Adapting an IT management framework described how to apply this approach to integrating data sources for use in web applications by taking an existing IT management framework and modifying it for the specific purposes of a data integration process. This could involve additions to the IT management framework to make it work.
for integration purposes. Contrast to the extension of a framework where the original framework remains as a base framework and the extensions form a new framework, adapting a framework involves working with just the base framework. Factors that can influence each IT management frameworks implementation includes being seen as complex, too general, and overlapping each other.

A software development framework is a software life-cycle process in the management process-based approach category. This category consists of a set of processes that contain procedures that can be used by managers for any type of software development project. This section described how adapting a software project management framework is an approach aimed at modifying an existing software development framework such as PRINCE2 to apply it as an approach for integrating data sources for use in web applications. This means taking an existing framework and adapting it for the specific purposes of integrating data sources for use in web applications. This would involve additions to the framework to make it work for integration purposes. Contrast to the extension of a framework where the original framework remains as a base framework and the extensions form a new framework, adapting a framework involves working with just the base framework.

Some criticisms of some of the examples from this approach are that they result in over responding to a change in requirements which has been cited as the source of many software disasters. The literature also suggests the more complex the system architecture and the software are, the more thinking behind the architecture that has to be done before coding can begin. The characteristics of current software development frameworks discussed in this section show that these frameworks, individually in their current form, are not appropriate as an approach for supporting all integrating scenarios for integrating data sources for use in web applications. They are however extensible so that they can be altered for the specific purpose of integrating data sources for use in web applications. They can be built on so that they can be applied for this purpose and used to support a data integrator in their integration process.

Toolkits are typically software interfaces designed to enable access to data in real-time. This section described applying this approach to integrating data sources for use in web applications involves modifying an existing development toolkit to suit the goals of
integrating data sources for use in web applications. This means taking an existing
development toolkit and enhancing it or modifying it for a specific data integration
process. In each example described, the tool kit is not web application focused and each
is a technology specific approach. Each of the toolkits discussed are designed to solve a
specific technical challenge and service a target technical environment. They are not
designed to be a generic approach that could be used across potentially vastly different
technical environments. Instead, each toolkit is single purpose specific.

Table 2.25 illustrates the different management process-based approaches in
comparison to each other against the classification criteria discussed.

<table>
<thead>
<tr>
<th>Model Type</th>
<th>Translation</th>
<th>Programmable or Fixed</th>
<th>Centralised or DeCentralised</th>
<th>Meta-data</th>
<th>Self Manageable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extending existing frameworks</td>
<td>Yes</td>
<td>Fixed</td>
<td>DeCentralised</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Adapting an IT management framework</td>
<td>Yes</td>
<td>Fixed</td>
<td>DeCentralised</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Altering software development frameworks</td>
<td>Yes</td>
<td>Programmable</td>
<td>DeCentralised</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Toolkits</td>
<td>Yes</td>
<td>Programmable</td>
<td>DeCentralised</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 2.25 Management process-based approaches comparison

At a high-level, there is a single overseeing framework that is still missing that can be
applied to these areas that could address the unique challenges identified within.
Extending existing frameworks or altering software development frameworks may serve
as the foundation to build on as a solution to this challenge.
2.7 Summary

This chapter reviewed the current state of research in the field of integrating data sources for use in web applications. This chapter provided a background to data integration, web applications and data sources investigating the approaches for integrating data from heterogeneous sources. It presented an investigation into the approaches and categorised them into three groups being model based approaches, technical approaches and groups of management process-based approaches. These three groups of approaches were evaluated by contrasting the characteristics of each against a classification scheme proposed in this chapter. It also reviewed the definitions of common terms and how data integration works between web applications and data sources. The examination of the characteristics of web applications discussed in this chapter help determine the viability of the model, technical and management approaches discussed for achieving successful data integration in all integrating scenarios.

This chapter discussed the literature relevant to the definition of data integration, web applications and data sources. It reviewed how these areas interrelate, their use and architectures. It provided the background necessary for understanding what data integration of data sources with web applications is all about.

It also presented an analysis of the literature on integrating data sources to be used with web applications discussing the complexity, heterogeneity, the process and the types of data sources. It examined the literature around defining the integration process focusing on the literature relevant to the three approaches to data integration being models approaches, application and management process-based approaches. Model approaches discussed the different groups of models such as layer models, common, schema, data mapping and language models. Technical approaches examined standardisations, brokers, extensible languages, exposed APIs, public APIs and extract transform loads. Management process-based approaches explored various software development processes, methodologies and frameworks such as extending existing frameworks, adapting a project or IT management framework or altering software development frameworks and toolkits.
A classification scheme was proposed and examined to demonstrate the applicability of the three groups of approaches for the purposes of presenting the data in a web application providing a means of integrating the data from the various sources in a consistent, timely and accurate manner. Each of the groups of models, technical and management process-based approaches discussed in this chapter demonstrated its validity and desirability in its respective application realm. Each featured a range of distinguishing characteristics that made it effective in achieving data integration for web applications. Many of the approaches are intended to support integration within a specific domain and were also found to have a series of difficulties associated with data integration as a result of contrasting the characteristics of each approach to the classification scheme discussed. Current approaches could be improved on from being domain and application specific, less complex to implement, and extended to avoid compatibility issues. They could be improved to support maturity and reusability of approaches and take into account new and emerging models and technologies to aid a data integrator in their integration process.

Both model and technical approaches examined were often purpose built for a specific application or domain, relied on strong technical reliance for maintaining and implementing the model and faced problems associated with time by handling the way data sources change over time. This was found in models across each of the different model groups including layer models, common, schema, data mapping and language models and technical approaches across the groups standardisations, brokers, extensible languages, exposed APIs, public APIs and extract transform loads. When examining the handling of data as it flows from one source to another, many approaches examined are able to handle this within their domain specific purposes but when requirements are outside the scope of the domain, consistency issues arise as seen with examples such as brokers, extract transform loads, language models and data mapping. In the event the destination and the source communicate differently to each other, this approach falls short.

The models and technical approaches discussed were all shown to be reliant on high-level technical implementation and maintenance. A non-technical data integrator would struggle to maintain and in some instances implement the models and technical approaches discussed without heavy reliance on an IT expert. Model and technical
approaches discussed were summarised as being domain specific and not appealing to all integration scenarios. They featured a range of distinguishing characteristics that made each an effective data integration implementation approach. But these characteristics were intended to support integration within a specific domain.

These approaches discussed are able to handle requirements within their domain specific purposes but when requirements are outside the scope of the domain, the difficulties associated with data integration discussed in this chapter arise. Technologies do exist to achieve integration, such as brokers and translations, but they cannot be used as a generic solution for developing web-applications.

The management process-based approaches discussed included approaches such as software development processes, methodologies and frameworks. Project management frameworks were shown to have difficulties in supporting the data integrator in an integration process. Data integrators are waiting for the data to be ready in the data warehouse before there are developed selected integration features which can result in days to weeks of delays. This can also result in data integrators who cannot wait for the data to be ready in the data warehouse and they develop their own solo integration solutions by going directly against the operational source systems. Frameworks with more complex system architectures resulted in more planning behind the architecture that has to be done before coding can begin. Agile methodologies do not recognize a service request for a new system to be the final set of requirements. These areas lead to the argument that they can be improved on to support the data integrator in their integration process.

Examining the extension of existing frameworks showed a series of difficulties where the framework is not a technical one. It involves founding a solution to integrating data sources for use in web applications on a framework not designed specifically to address this technical challenge. When examining IT management frameworks, the body of literature showed factors that can influence each frameworks implementation included being seen as complex, too general and overlapping each other.

The areas for improvement with existing model, technical and management process-based approaches discussed in this chapter highlights the challenges currently faced
when attempting integration of data sources with web applications. This results in the need to improve these areas to support a data integrator in their integration process. It leads to the viability for an approach to build on the existing solutions for supporting a data integrator by improving their data integration process when integrating data sources for use in web applications. Attempting a data integration process for use in web applications using existing approaches could be extended to support a greater range of integration technologies. The literature examined supports the argument for the need for an approach to take into account new and emerging models and technologies and be applicable for the maturity and reusability of existing technologies for data integration. It needs to overcome compatibility issues, reliance on a technical expert and high cost to maintain. It can be used to consult with and aid in attempting to integrate with data sources for web applications and enable the data integrator such as a web developer to choose the most appropriate integration approach to implement.

The viability of the current approaches discussed in this chapter does not appear appropriate for supporting the data integrator in all integrator scenarios. This chapter has examined a pattern of difficulties with the different groups of approaches.

Amongst the different model, technical and management approach groups, there are a common group of areas that could be improved:

- Intended to support integration within a specific domain
- Approach is too complex
- Compatibility issues
- Lack of maturity and reusability
- Approach is too general and overlaps other approaches
- Doesn't take into account new and emerging models and technologies

Table 2.26 shows the pattern of difficulties across the different model approach groups discussed in this chapter, evaluated by the classification scheme.
<table>
<thead>
<tr>
<th>Model Type</th>
<th>Translation</th>
<th>Programmable or Fixed</th>
<th>Centralised or DeCentralised</th>
<th>Meta-data</th>
<th>Self Manageable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Data Model</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Data Mapping</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Language Models</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Schema Mappings</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Meta-Tagging Model</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Layer Models</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 2.26 Classification scheme overview of model groups

Table 2.27 shows the pattern of difficulties across the different application approach groups discussed in this chapter, evaluated by the classification scheme.
<table>
<thead>
<tr>
<th>Model Type</th>
<th>Translation</th>
<th>Programmable or Fixed</th>
<th>Centralised or DeCentralised</th>
<th>Meta-data</th>
<th>Self Manageable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standardisations</td>
<td>Yes</td>
<td>Fixed</td>
<td>Centralised</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Brokers</td>
<td>Yes</td>
<td>Fixed</td>
<td>Centralised</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Extensible Languages</td>
<td>Yes</td>
<td>Programmable</td>
<td>DeCentralised</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Exposed APIs</td>
<td>Yes</td>
<td>Programmable</td>
<td>DeCentralised</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Public APIs</td>
<td>Yes</td>
<td>Programmable</td>
<td>DeCentralised</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Extract Transform Load</td>
<td>No</td>
<td>Fixed</td>
<td>DeCentralised</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 2.27 Classification scheme overview of application groups

Table 2.28 shows the pattern of difficulties across the different management process-based approach groups discussed in this chapter, evaluated by the classification scheme.
Table 2.28 Classification scheme of management process-based approaches

The viability of applying the existing model, technical approaches and management process-based approaches to achieving integration of data sources for the purposes of presenting the data in a web application on a wide high-level scale is quite low. There is therefore a need to develop a solution to improve the process of integrating data sources for the purposes of presenting the data in a web application providing a means of integrating the data from the various sources in a consistent, timely and accurate manner.

A solution should support the classification scheme discussed in this chapter. The solution would support the technical approach chosen for the integration process acting as a conceptual structure intended to serve as a support or guide for the integration process. This would enable the adoption of specific technical practices for the implementation of the integration process itself. It could also be applied in all integrating scenarios. This will lead to the ultimate goal where the challenges discussed in this chapter are more easily managed during an integration process.
3 Supporting the integration of data sources for use in web applications

3.1 Overview

The aim of this thesis is to support a data integrator such as a web developer in improving their data integration process when using multiple data sources in a web application. Chapter 2 argued that there is a lack of generic supporting architectures that can be applied to a range of integration scenarios. A data integrator has a complex decision-making process to go through to determine which integration approaches are the most appropriate (see Section 5.2.2). There are three styles of approaches, management process-based approaches, technical approaches or data integration models (see Sections 2.4, 2.5 and 2.6), and it is not always possible to choose from these approaches. To be applicable to any particular scenario, these approaches need to be combined or modified. Attempting an integration process between data sources for use in web applications using existing approaches has been shown to be problem or technology specific. The difficulty is that due to the lack of a generic framework, current approaches cannot always take into account new and emerging models and technologies (see Sections 2.4.9 and 2.5.9). They can be broadly domain specific which may result in the data integration outcome having inconsistent reliability due to compatibility issues (see Sections 2.4.9 and 2.5.9), they are reliant on an Information Technology (IT) technical expertise and can have a high cost to maintain. This can lead to failing the data integration goal. Choosing an appropriate data integration approach and technology is an important step in the web development process which requires deep knowledge of many possible approaches and technologies. The consequences of choosing the wrong approach or technology could result in the data integration failing or only achieving a partial success, poor quality data results or incorrectly matched data fields (see Section 2.5). The data sources (such as web services or schemas) that web applications rely on are often structured differently from one another and are difficult to integrate (see Section 2.5).
This chapter argues for the need for a generic approach and presents an investigation into the solutions for supporting a data integrator when integrating data sources for use in web applications. This chapter presents an investigation into the need for a solution that helps a data integrator address the challenges discussed and defines the requirements for achieving this. Subsequent chapters will explore the application of these requirements in web developer cases (see Chapter 5). These cases explore different spectrums of web development at a theoretical level and do not involve actual web developers which is a limitation of this study.

This remainder of this chapter is organised as follows. Section 3.2 argues for the need for a model to support a data integrator when integrating data from data sources for developing web applications. Section 3.3 argues for a framework that will be used by developers in creating web applications using data from multiple data sources. Section 3.4 presents an investigation into the need to address the challenges for web application development when integrating data between data sources and web applications using existing approaches (see sections 2.4, 2.5 and 2.6) to improve the area of web application development. Section 3.5 describes the list of requirements for a supporting model that allows a data integrator to integrate data sources for use in a web application. Section 6 presents a summary of this chapter.
3.2 The requirement for supporting the integration of data sources for use in web applications

There is a need to support the integration of data sources for use in web applications. Figure 3.1 illustrates the workflow of a data integrator for web development using a UML Component Diagram detailing how the integration approach chosen relates to the web application development process.

![UML Component Diagram Integration Technology Decision in the Web Development Process](image)

Figure 3.1 UML component diagram integration technology decision in the web development process

Figure 3.1 illustrates where the decision on which integration technology to choose as an approach for the integration process sits in the web development process. There is very little documentation of problems faced by practical data integrators in the literature. For the purposes of this work, a typical web development process will need a decision on which integration approach to use and subsequently the web developer implements this integration technology. This then leads to the implementation of the integration development.
Building on this workflow, Figure 3.2 shows a UML Activity Diagram illustrating the process the data integrator goes through when implementing a chosen integration technology. It highlights several outcomes a data integrator has to overcome or workaround in the event their chosen integration process results in those outcomes.

Figure 3.2 illustrates how a web developer who follows the figure left to right runs into various problems as part of a typical data integration process workflow as described in this work. The web developer must choose an integration path from a large number of alternatives. The outcome is not always clear at the time of making the choice, potentially resulting in failure. The viability of the different integration approaches is potentially unknown to the web developer in relation to the characteristics of their web application. This can result in the problems discussed (see Section 2.4 and Section 2.5) being encountered by the web developer and subsequently a failure in their integration process or an integration result that is not desirable or well supported for the long term. This could occur, for example, when the integration technology chosen may have a high cost to maintain or it may have compatibility issues.

A development solution which can support the data integrator in many different integration scenarios is desirable. It would need to be applicable in many different
integration scenarios covering a wide range of web applications and their associated data sources meaning that the development solution needs to be more generic in its aims. A development solution would also need to be able to take into account new and emerging integration models and technical approaches. It would need to act as a guide for a web application builder to consult with as part of their integration process such as integrating an application programming interface (API), web service or schema.

A generic approach needs to have selectable integration options from which a web developer could choose from based on the characteristics of their web application (see section 2.3.3). It encompasses the decision making process (see section 2.6) and will need to act as a workflow to guide a web application builder in mapping out an integration path and choosing the right integration technologies (see section 2.5) to implement in their integration process. For example, a web application builder could consult the approach to establish what workflow to implement for their integration process by examining the characteristics of their web application to map out the integration path to take and the subsequent integration model and technical approach to follow.

3.3 The need for a framework

3.3.1 Overview

This section argues that the most appropriate way to support the integration of data sources for use in web applications is with an architecture based approach such as a management framework. A methodology would need to be instantiated through a framework. A framework used to assist and support the development that facilitates the different technologies. The methodology is a data integration specific methodology rather than a generic IT management methodology that bridges the gap between the technology and process approaches.

3.3.1 The need for a framework

The need for a framework for supporting the integration of data sources for use in web applications stems from a need to improve on existing approaches as discussed in
Chapter 2 (see sections 2.4, 2.5 and 2.6). These include improving on a lack of generic frameworks that take into account new and emerging integration models and technologies, a lack of maturity and reusability of current approaches, and compatibility issues where approaches are domain or problem specific. There is a need to improve these areas to improve web application building involving integrating data sources.

Figure 3.3 shows a UML Activity Diagram of the current workflow a data integrator such as a web developer goes through in making a decision as to which integration path to take. In this workflow they consider existing models or technology approaches.

In Figure 3.3, the integration outcomes success depends on a degree of risk in during the decision making phase on this workflow. There is no generic framework or guide that is supporting their decision on which model or technology approach to utilise. The resulting integration outcome may not have been the ideal or correct path to follow.

A workflow for developing dynamically integrating applications would help formalise and standardise procedures that a data integrator could follow for their data integration process (see section 2.6). The most appropriate way of doing this is to develop a support framework that facilitates the different technologies. This approach is a high-level architecture based approach. It’s a data integration specific methodology instantiated through a framework rather than a generic IT management methodology.
bridging the gap between the technologies approaches and processes. Extending an existing framework would not be appropriate for a generic high-level application to many different integration scenarios (see section 2.6). Using existing model of technical approaches (see section 2.4 and 2.5) cannot always take into account new and emerging models and technologies (see Sections 2.4 and 2.5). They can be broadly domain specific which may result in the data integration outcome having inconsistent reliability due to compatibility issues (see Section 2.4), they are reliant on an Information Technology (IT) technical expertise and can have a high cost to maintain.

A framework is needed that is geared towards supporting a web application builder in their integration activities (see section 2.7). It would act as a generic framework that could be applied to many different integration scenarios. A framework would not be problem or domain specific which is a shortcoming of current approaches. Instead it acts as a guide for a data integrator such as a web application builder to consult with for their integration process regardless of its nature and facilitates different technologies. A framework needs to improve the data integration process for the purposes of presenting the data in a web application providing a means of integrating the data from the various sources in a consistent, timely and accurate manner. Figure 3.4 shows an UML Activity Diagram that illustrates how the framework fits in the workflow a web developer goes through in an integration process.

Figure 3.4: UML activity diagram showing support framework that can be applied to different integration scenarios and facilitate different technologies
Figure 3.4 highlights a support framework enabling a data integrator to be guided by the framework in choosing the optimal integration approach facilitating different integration technologies.

3.3.2 Defining requirements for integration processes

To develop a support framework to mitigate the problems discussed in Chapter 2 (see sections 2.4, 2.5 and 2.6), there is a need to define both technical requirements and user requirements for achieving this aim. Technical requirements consist of the technical outcomes that need to be supported by the approach. User requirements provide for a user who is attempting a web application data integration process. Establishing requirements will help formulate how to develop a framework to integrate data sources for use in web applications.

The requirements will ensure that the framework proposed in this thesis supports the solution chosen for the integration process by a data integrator. The requirements are a core set of goals that the framework must support. These requirements should appeal to what a user who is attempting an integration process would expect to see and system requirements which are technical outcomes that need to be supported. The requirements validate the conceptual structure intended to serve as a support or guide for the integration process. The requirements also help guide the technical areas in the implementation of the data integration process.

3.4 Approach for supporting the integration of data sources for use in web applications

This section argues for the need for an approach to the problems discussed in Chapter 2 (see sections 2.4, 2.5 and 2.6) and the requirements for achieving this aim. It describes how an architecture based approach would support a data integrator with a better framework, choosing a methodology and choosing different frameworks.
As discussed in earlier sections (see Sections 2.4, 2.5 and 2.6), there are challenges for web application development when using existing approaches and there is need to address these challenges with an approach to add to the area of web application building. A workflow for developing dynamically integrating applications would help formalise and standardise procedures that a data integrator could follow for their data integration process. The most appropriate way of doing this is to develop a support framework approach.

3.4.1 A methodology instantiated with a framework

An approach to deal with the challenges discussed in Sections 2.4, 2.5 and 2.6 is a generic integration methodology instantiated through a support framework. It is for a data integrator such as a web application builder to use for their data integration process with web applications. The methodology sets out the process to follow for the data integration process instantiated through a framework which will act as a guide for a data integrator to follow.

A framework would enable the data integrator to choose the most appropriate integration model, process or technology to use based on examining their integration activities and challenges, consulting the framework and resulting in making an informed choice about which integration path to take.

A framework acts as a guide enabling a data integrator such as a web application builder to examine their web application and compare its characteristics using the framework to determine which integration path to follow. Figure 3.5 illustrates a UML Activity Diagram of how the framework supports the data integrator in the classification of their application to determine which integration path is the most appropriate to take.
In Figure 3.5, the data integrator follows a methodology to classify their type of web application and utilize the framework to determine the integration pathway to follow. This path could include a mixture of integration models and technologies to use for their integration process. There is a need to identify requirements for this approach which are discussed in section 3.5.

3.5 List of requirements

3.5.1 Overview

This section introduces a set of requirements for the proposed approach. It discusses how the requirements are derived and explains what these requirements are.

3.5.2 How requirements are derived

This section examines the difficulty of anticipating and meeting all data integration requirements across an organisation or integration scenario. When integrating web applications, understanding the nature of the problem can be very difficult. It may be difficult to establish exactly what the system should do (Sommerville, 2001). When examining what is required for a data integration process to be successful, the number of factors that must be considered depends on the technical environment of the
implementation process and its intended outcomes. From a generic standpoint, an application needs specific data at a given time, in a specific format and in a required quality (Schwinn & Schelp, 2005). The requirements for achieving a successful data integration outcome all serve this need.

A list of requirements will determine what a framework for data integration should do to as well as formalise what requirements can be followed in a data integration process to identify what is needed to successfully achieve a data integration goal. Requirements will aid in understanding the architecture of the data integration process, the goals and boundaries to achieve a successful data integration outcome and the expected performance and support that is required. Figure 3.6 presents a UML Component Diagram of how at a high-level how this workflow connects with the web development process for an integration activity. It illustrates where the requirements identification stage sits in the web development process.

Figure 3.6 UML component diagram of requirements identification in the web development process
Figure 3.6 shows that the requirements aspect of an integration workflow sits right at the beginning as the starting point.

Two groups have been identified as common foundations for requirements. One is user requirements meaning the high-level abstract requirements in natural language and diagrams of what services the system is expected to provide and the constraints under which it must operate (Sommerville, 2001) and the other is system requirements meaning the detailed description of what the system should do setting out the system services and constraints in details, often presented through a functional specification (Sommerville, 2001). Both of these requirements areas are relevant to the data integration process as the requirements for data integration of data sources with a web application with must have consideration for both of these target areas.

Errors in the requirements can lead to extensive rework costs (Beck, 1999), (Sommerville, 2001). The physical cost and the cost in time can be enormous when half way through the implementation phase it is realised that the requirements that the integration process is based off were incorrectly specified. The repair task could result in having to start over from scratch or investing a large quantity of unanticipated hours in extra and unnecessary development. It is important to ensure that the requirements define the system which the user actually wants (Beck, 1999; Sommerville, 2001) and this is not misjudged.

In Sommerville’s requirements validation process, different types of checks are carried out on the requirements. Validity checks compare what is thought to be the functions and tasks that are needed with the results of analysis to identify additional or different functions required (Sommerville, 2001). This is especially relevant when there are different types of end users for a system and only one user type is being considered in an integration requirements analysis. Consistency checks to ensure requirements do not conflict with each other (Sommerville, 2001). Completeness checks ensure the requirements include and define all functions and constraints intended by the user (Sommerville, 2001). Realism checks ensure the requirements can actually be implemented (Sommerville, 2001). Beck suggested collecting user requirements incrementally and then prioritising requirements for implementation (Beck, 1999). The requirements should be evaluated for the cost of implementing them, the existing and available technologies, the knowledge required and any potential constraints.
Requirements validation is an integral checking step in identifying requirements in that system.

### 3.5.3 Requirements for supporting the integration of data sources for web applications

A list of requirements is broken up into two groups being the user requirements called integrator requirements which are the requirements that a data integrator who is attempting an integration process would expect to see and system requirements which are technical outcomes that need to be supported. Section 3.5.3.1 presents the Integrator requirements and Section 3.5.3.2 the System requirements. These two groups were derived from Section 3.5.2 where Sommerville (2001) had identified them as common foundations for requirements. When considering requirements for supporting the integration of data sources for web application, Sommerville and Beck identified checks and validation needed in requirements and how these checks are derived. In the context of integrating with web application data sources, characteristics such as the consistency of the integration method, the timeframe is takes to complete the task and accuracy of results are relevant user checks. Reliance on deep knowledge experts to implement the task is also an important user validation check. Systematic characteristics such as the design, heterogeneity, structure and performance are checks that are applicable for validating these elements of the actual application itself.

#### 3.5.3.1 Integrator requirements:

Table 3.1 presents the Integrator requirements for a data integrator user such as a web developer.
1: The process of integrating data sources for the purposes of presenting the data in a web application must provide a means of accessing the data from the various sources in a consistent, timely and accurate manner.

2: The ability to implement, manage and control the integration process without the ongoing reliance on a third party or highly skilled technical person

Table 3.1 Integrator Requirements

3.5.3.2 System requirements:

Table 3.2 presents the system requirements consisting of the technical outcomes that need to be supported.
• 1: Design Process

The integration process must describe the architecture being integrated with the goal of modeling the data integration pathway to be implemented. This will aid data integrators in mapping out the right path to follow and providing a structured design process for the data integration process. As part of identifying and modeling the integration landscape, validating the technical requirements for the implementation of the integration process is also integral.

• 2: Heterogeneity

The integration process must support heterogeneous environments, integration from a variety of sources consisting of different structures and formats from mainframes to messaging systems. This could include different file formats, access protocols and query languages, different ways of representing and storing the same data and even relationships between data.

• 3: Structure and Integrity Support

The integration process must support data quality and data governance. The data translated or transitioned from one source to another must not lose its integrity. Its format may change from one source to the other as part of the integration process but the information contained should be presented the same without any loss or misinterpretation from what as in the original source.

• 4: Performance

The integration process must ensure the data integration process maintains optimal performance when integrating between a data source to its destination. There should be reduced lag in presentation or reduced loss in quality of data. The framework should have the ability to recover from errors such as source or
target systems going down and data quality issues.

Table 3.2 System Requirements

Table 3.2 sets these requirements out as a formal checklist that can be consulted with as part of evaluating the integration process requirements.

3.5.4 Integrator requirements:

The integrator requirements are from the perspective of a user such as a web developer who is attempting a data integration process. It is what they would expect as an ideal outcome from the process. Each of the components of this requirement forms the expectations of the user’s ideal outcome. In this work, the user is a ‘data integrator’ who can be a web developer, blog builder, or middleware programmer bringing together various forms of data to make a smoothly functioning application. The user wants to achieve live data access with their target sources, they want to achieve this in a fast and accurate manner. They have skills in developing a web application which could involve using a third party authoring tool such as Wordpress or writing programming code from scratch. Each of these areas relates back to the common characteristics discussed in Table 3.2 with identification of requirements and it is these characteristics combined into this user requirement that form the basis of what ideal scenario should be achieved from a successful data integration process between web applications.

This data integrator requirement should be consistent across all integration scenarios and each of the boundaries tested against as a means of ensuring the data integration process is successful or not in each area. If the integration process fails in one of these areas, then the requirement is not being met. For example, integration may be consistent and accurate between target sources but not fast therefore meaning the data integration is not in real time and could result in delays in presenting data.

It is important for the data integrator to implement, manage and control the integration process without the ongoing reliance on a third party or highly skilled technical person.
Reliance on technical personnel to perform these tasks for you can be costly in ongoing maintenance. They can also be limiting in the expansion of web applications since a technical expert would be required for any additional integration objectives.

Assisting the data integrator in generating connection specific code for different data sources would reduce the reliance on a technical expert. A development tool such as Eclipse that a data integrator uses to develop their web application is extendable to be able to generate connection code for different types of data sources to establish a connection with them. This might also involve expanding the current data source selection options.

Characteristics the data integrator should find from their integration process should include the characteristics show in Table 3.3.

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ease of use of the solution</td>
<td>The data integrator finds the solution easy to implement</td>
</tr>
<tr>
<td>Architecture</td>
<td>The solution is backed by an architecture</td>
</tr>
<tr>
<td>Testability</td>
<td>Where the integration process can be tested for measurable outcomes</td>
</tr>
<tr>
<td>Support</td>
<td>Where the integration process is backed by support ranging from documentation to a knowledge base or other support resources</td>
</tr>
<tr>
<td>Error handling</td>
<td>The integration process is flexible for handling unpredictable errors</td>
</tr>
<tr>
<td>Monitoring</td>
<td>The ability to monitor the data throughout the integration process</td>
</tr>
<tr>
<td>Readiness</td>
<td>The readiness of the solution to implement in an integration process.</td>
</tr>
</tbody>
</table>

Table 3.3 Data integrator requirement characteristics
It is the characteristics of these requirements that will contribute to evaluating the framework proposed in this thesis.

3.5.5 System requirements:

These requirements consist of the technical outcomes that need to be supported. They are described in four parts as previously defined in Table 3.2, Design Process, Heterogeneity, Structure and Integrity Support and Performance with each of these requirements focusing on the technical considerations in each area.

3.5.6 Design process

As described in section 3.4, the integration process must describe the architecture being integrated with the goal of modeling the data integration pathway to be implemented. This would identify the requirements for initiating and achieving a data integration outcome, providing a structured design process to aid the data integrator in developing a data integration process. Without it, the integration process could fail because it didn't address the known challenges associated with different architectures.

This requirement will ensure the identification of the overall design of the integration process from start to finish. It helps create a full picture of how the integration outcomes are going to be achieved. Validation of the technical requirements is integral to the success of the identification and modeling of the integration process. Validation involves examining the consistency and completeness of technical requirements for the implementation of the integration process and validating their needs. By outlining the integration landscape, differences between the source and the destination can be identified and the platform for achieving the integration process is fully understand. All of these areas combined are integral for achieving a successful data integration outcome.
3.5.7 Heterogeneity

The integration process must support heterogeneous environments. The framework must support integration from a variety of sources of different structures and formats. This could include different file formats, access protocols and query languages, different ways of representing and storing the same data and even relationships between data. If a data integration process cannot support integration of every data source, then some information will be lost or is unable to be integrated with. This requirement will ensure the integration process supports all these different data sources and is necessary for achieving a successful data integration outcome.

3.5.8 Structure and integrity support

The integration process must support data quality and governance. The data translated or transitioned from one source to another must not lose quality such as degradation to its structural integrity. This requirement is necessary for successful data integration outcomes. As previously discussed, when encountering different format types, the data is not always necessarily translated from one source to another so clearly and precisely.

Without this requirement, an integration process may fail to integrate important information or lose it altogether as data is transformed between data sources thereby leading to a total failure of the integration process. This requirement will ensure that the data integration process maintains the integrity of the data being integrated. There cannot be any loss of integrity between sources. Its format may change from one source to the other as part of the integration process but the information contained should be presented the same without any loss or misinterpretation from what as in the original source.

3.5.9 Performance

The integration may be consistent and accurate between target sources but not fast resulting in delays in presenting data. The integration process must ensure the data
integration process maintains optimal performance when integrating a data source to its destination at a technical level as well. There cannot be any lag in terms of required and expected performance in presentation or loss in quality of data.

Performance doesn't just mean speed but also the ability to recover from errors such as source or target systems failure or data quality issues. In the event of an outage of a target data source, the integration process should be able to look at other sources for the data or have a plan for ensuring and handling uptime so that the rest of the integration process does not suffer as a result of an outage from one target data source, a process known as performance transparency. System characteristics that should be supported include the characteristics shown in table 3.4.

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer centric versus designer centric</td>
<td>The integration tool or solution should have an appeal to both the developer and designer</td>
</tr>
<tr>
<td>Expandability</td>
<td>The integration tool or solution should be expandable for future development requirements or evolution in requirements</td>
</tr>
<tr>
<td>Deployment</td>
<td>It should have an easy deployment process for the data integrator to follow</td>
</tr>
<tr>
<td>Popularity</td>
<td>It should be seen as a popular choice by web developers and other data integrators for a particular integration scenario or activity</td>
</tr>
<tr>
<td>Tool support</td>
<td>The integration tool or solution should have a meaningful support or knowledge base such as strong documentation</td>
</tr>
<tr>
<td>Connectivity</td>
<td>The integration tool or solution should be connectable with other technologies</td>
</tr>
<tr>
<td>Domain specific language</td>
<td>Although many integration tools and solutions will be created in domain specific language, it should be open for being used</td>
</tr>
</tbody>
</table>
3.5.10 Requirements a framework must support

It is the characteristics of all these requirements discussed above that will contribute to evaluating a framework for supporting the integration of data sources. Table 3.4 summarises these characteristics.

<table>
<thead>
<tr>
<th>Design Process</th>
<th>Heterogeneity</th>
<th>Structure and Integrity Support</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Where does the Integration task or process fit?</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.5 Requirements for data integration

A framework for supporting integration of data sources must support the four requirements of Design Process, Heterogeneity, Structure and Integrity Support, and Performance. It must support them to act as a support framework that could be used in all data integration scenarios for integrating with web applications.

A framework must describe the architecture being integrated with the goal of modeling the data integration landscape to be implemented. This would identify the requirements for initiating and achieving a positive data integration outcome, providing a structured design process to aid the data integrator in developing the most appropriate data
integration process. It will also validate the technical requirements as part of this requirement. A framework must support heterogeneous environments. The integration process must support data quality and data governance as part of the Structure and Integrity Support requirement. Finally, it must support Performance to ensure the data integration process maintains optimal performance when integrating between a data source to its destination at a technical level as well.

3.6 Summary

This chapter examined how the current approaches discussed in Chapter 2 have a common pattern of difficulties associated with data integration between the groups of management process-based approaches, model and technical approaches. It discussed how these approaches are problem or technology specific and cannot be used as a generic solution in all integration scenarios. It discussed the need to improve on a lack of generic frameworks that also take into account new and emerging models and technologies, a lack of maturity and reusability of current approaches, compatibility issues, reliance on an IT technical expertise and high cost to maintain. The chapter then argued that there is a need to address these problems to improve the integration of data sources for use in web applications. It argued that there is a need for a better approach to aid a data integrator such as a web application builder in their integration process.

This chapter argued that this would be achieved through a technical solution such as a management framework. A support framework helps avoid these problems and defined the requirements for achieving this aim. It discussed the need to develop a framework to integrate data sources for use in web applications as a solution and discusses the requirements for this approach. It introduced an approach for supporting data integration with data sources for web applications. It examined the requirements for this approach to support the integration of data sources for web applications. It investigated the requirements that this approach should support. It also looked at the definitions behind requirements, why they are needed and how they are derived. It examined the different approaches to the discovery of data integration requirements contrasting the characteristics of each to derive a core list of requirements for the proposed framework of this thesis.
This chapter described how this thesis will address these problems described with the approach proposed and the requirements for achieving this aim. It described how an architecture-based approach would support a data integrator with a better framework, choosing a methodology and choosing different frameworks. The proposed approach is an integration methodology instantiated through a support framework. It is for a data integrator such as a web developer to use for their data integration process with web applications. The methodology sets out the process to follow for the data integration process instantiated through a framework which will act as a guide for a data integrator to follow.

It also presented how requirements are identified and derived with a list of requirements that the framework must support. A list of requirements is presented that determine what a framework for data integration should support as well as formalise what requirements can be followed in a data integration process to identify what is needed to attempt and successfully achieve a data integration goal. Requirements will aid in understanding the architecture of the data integration process, the goals and boundaries to achieve a successful data integration outcome and the expected performance and support that is required. These requirements are broken up into two groups, integrator requirements which are the requirements that a user who is attempting an integration process would expect to see and system requirements which are technical outcomes that need to be supported. It is the characteristics of these requirements discussed that will contribute to evaluating the framework proposed in this thesis. The framework must support the Design Process, Heterogeneity, Structure and Integrity Support and Performance.

The derived framework requirements discussed in this chapter are an integral contribution to the research questions posed in this thesis. It contributes directly to the second research question posed in this thesis: What characteristics of web applications determine what integration method to use for integrating with data sources. The requirements will ensure that a support framework supports the solution chosen for the integration process acting as a conceptual structure intended to serve as a support or guide for the integration process. The requirements contribute to the characteristics that determine what integration method to use for supporting the integration of data sources for web applications. They contribute to validating and evaluating a methodology and
framework to determine its applicability to support integration of data sources for web applications in all integration scenarios.
4 GIWeb: An approach for supporting the integration of data sources for use in web applications

4.1 Overview

Chapter 3 argued for the need for a generic approach for the integration of data sources for use in web applications. This chapter introduces an approach called GIWeb to support the data integration process for a broader set of integration goals than the domain specific objectives of current approaches. It aims to bridge the gap between integration technologies and processes, distinct from generic IT management frameworks through an integrated technology and model approach.

GIWeb builds on the existing solutions by improving their data integration process. It takes into account new and emerging models and technologies yet still applies to maturity and reusability of reusable technologies. It attempts to overcome compatibility issues, reliance on a technical expert and high cost to maintain (see sections 2.4, 2.5 and 2.6). It enables the web developer to choose the most appropriate integration approach. GIWeb is a goal centric approach with a top-down design. Data integration involves working out the goal first, identifying where the web application fits, applying a methodology and importing the technology to apply it followed by a testing element.

The remainder of this chapter is as follows. Section 4.2 introduces the proposed methodology and outlines GIWeb. It shows how it will support a data integrator in their integration process. It discusses how it supports the requirements presented in Chapter 3. GIWeb is a proposed framework and in Section 4.3 an instantiation of it is demonstrated. Section 4.4 presents a summary of this chapter. Examples of how GIWeb is implemented are discussed through cases in Sections 5.2, 5.4, 5.6 and 6.3.
4.2 Methodology

4.2.1 Overview

A systematic development process can use standard or company specific frameworks, methodologies, modeling tools and languages (Alexander, 2014). The steps that are implemented in the process vary between project projects but do follow a similar pattern. These steps include: review, assessment and analysis, specification building, design and development, content writing, coding and testing (Alexander, 2014). Steps have also been defined as including a roadmap document defining the web application, purpose, goals and direction (Kohan, 2014). A technology selection step is either a standalone step or is part of a of the specification process. It is this phase of the web development process that the methodology aims to support the web developer when integrating their web application with data sources. To illustrate with an example, a data integrator such as a web developer uses an IDE such as Eclipse to develop their web application. They use a model such as PRINCE2 to guide them. The proposed methodology aims to be pluggable into each of these development tools to aid the web developer in their integration process.

As argued in Chapter 2, to successfully integrate data for large applications, web developers need to adopt a disciplined development process and a sound methodology. The proposed methodology formalises and standardises procedures that a data integrator could follow (see section 2.7). This acts as a workflow. This could be supported in the development environment through the use of a workflow tool or wizard which would take the developer through the process step-by-step.

Figure 4.1 illustrates a UML Component Diagram of the high-level view of the components of the proposed methodology and the related workflow a web developer would follow. A web developer compares the characteristics of their web application to work out which integration path to follow. From this they can choose the appropriate models and / or technical approach to implement for their integration process.
4.2.2 Development of a methodology

The proposed approach is named Generic Integration for Web Applications (GIWeb). This section will introduce a high-level overview of the methodology component of GIWeb with section 4.2.5 to discuss how it is administered and implemented. The methodology is provided in the following two supporting forms.

- A diagram outlining the process methodology as a whole
- A spiral-model of the process methodology to illustrate the different iterations in each life cycle

The methodology will act as an organised, documented set of procedures that can be used step by step to carry out the intended integration. It will also act as an objective set of criteria for determining whether the results of the procedure are of acceptable quality. It will help determine goals, motivation and purpose for the integration task and allow an integrator to assess where their application fits and what integration process to follow.
The integration of data sources for use in web applications has many consequences and the risks of improper data integration are well understood in enterprise systems development (Azadeh, Saberi, Ghaderi, Gitiforouz, & Ebrahimipour, 2008). It requires a deep knowledge of many possible approaches and technologies (see Sections 2.4, 2.5 and 2.6). Choosing the wrong approach or technology could result in the web developer having to start over and try a different technology, increased cost or total cost blowout to the project, a large amount of wasted time or even a total failure in the project outcomes (Laudon, & Laudon, 2004; Mork, Halevy, & Tarczy-Hornoch 2001). The proposed methodology aims to simplify this by dividing the integration process into structured phases that can be applied sequentially or individually, according to the integration activity. The methodology is divided into six phases.

- 1 Analysis phase
- 2 Classification phase
- 3 Development phase
- 4 Test phase
- 5 Deployment and Release phase
- 6 Quality phase

Prior to the phases are a series of pre-initiation activities. The purpose of the pre-initiation activities is to build the foundations of an integration activity. The aim is for a web developer to identify what is needed for the phases of the methodology. The pre-initiation activities are represented in a UML Component Diagram as illustrated in Figure 4.2.
The methodology phases stem from the pre-initiation activities. Figure 4.3 illustrates a UML Component Diagram of the workflow relationship between each of the phases. Each relationship is discussed further below.
Figure 4.3: UML component diagram of the methodology for supporting the integration of data sources for web applications.

Figure 4.4 illustrates a sequence of phases from the methodology.
The following sections describe each of the phases in the methodology and their relationship to each other.

4.2.3.1 Analysis phase

In the analysis phase, the data integrator determines the goals, motivation and purpose of their data integration task. The analysis phase is where the data integrator questions what the specific purpose of the web applications are and the need to integrate the data sources between them. Determining the motivation involving asking if the data integration task is for solving a specific problem or making an operation more effective. This forms the problem definition. From this, it can be determined if all the problems can be solved with the intended data integration task.
The analysis phase is broken up into the following activities:

- Collecting user requirements and the data needed to present to the user
- Assigning priorities to each requirement
- Determining data sources to be integrated such as data through APIs or plugins
- Determining which web applications being integrated
- Conceptual design by the data integrator of the overall data integration architecture between the different web application data sources and web applications being integrated
- Determine technical environments of each web application and data source involved

Once this phase has been completed, it leads directly to the next phase of the methodology: Classification phase. Figure 4.5 illustrates this workflow between the different activities described with a UML Component Diagram.

Figure 4.5: UML component diagram analysis phase of workflow activities to classification
Figure 4.5 highlights the relationship between the different analysis phase activities. Determining goals, motivation and purpose of the integration activity is followed by defining the problem leading to the integration method and classification. It’s from the analysis phase that the planned outcome is determined and measured against in the later testing and quality steps.

### 4.2.3.2 Classification phase

The classification phase is the next activity of the methodology where the web application involved in the integration process is categorised based on their characteristics against a classification scheme to determine which data integration process should be implemented. This phase acts as a set of guidelines for the data integrator to use to determine and map out which path to take for their data integration plan.

The classification phase consists of several parts. This begins by determining the web application type by comparing the characteristics of the web application being integrated. Once the web application type is determined with a classification category, the data integrator can then look at what data integration path to follow from similar web applications. Contrast with choosing the wrong data integration path because the approach suits a different web application type to the one the data integrator has.

The classification scheme introduced in chapter 2 was used to examine the characteristics and boundaries of the web application and its data sources to categorise which web application type the application belongs to. These dimensions the web application is tested against are as follows:

- Data Translation dimension
- Programmable dimension
- Centralised dimension
- Meta-data dimension
- Self-manageable dimension
Once the dimensions are used to discover where the web application fits, its integration characteristics become clear. The web application can now be categorised based on these dimensions to determine its web application type.

4.2.3.3 The development phase

The development phase is where the data integration task is developed at a technical level. This involves the development of the data integration method to integrate the different data sources and the web applications involved.

This phase consists of two components, the logical design and architecture of a web-based integration project. The first component involves activities such as defining the selection of an XML standard, data mappings and schema definition. The second component involves architecture analysis such as the selection of an integration interface, API, error handling techniques, monitoring methods and logging.

Figure 4.6 illustrates a UML Component Diagram showing the development phase activities.
Figure 4.6 shows the workflow between the two development phase activities: Logical design and Architecture analysis. The Logical design activity consisting of defining activities such as defining XML standards, data mappings and schema definitions. Architecture analysis consisting of selecting an integration interface, error handling techniques and logging and monitoring methods.

Depending on the type of web application and data sources involved, the development phase typically involves a level of programming and database configuration by the data integrator. The data sources could range from XML schemas, dynamic database connections, APIs, data mappings or web services, each with its own technical methods for integrating with them which the data integrator will have to develop. The outcome of the classification phase helps the data integrator to determine which technical method to implement for the data source integration (see sections 4.2.4 and 4.3).

Once the development phase is complete, the test phase follows to ensure it all works as intended and to achieve the expected outcomes from the analysis phase.
4.2.3.4 The test phase

The test phase is where the activities from the development phase are validated to ensure they work as intended. The validation checks and measures the outcome result of the test against the planned outcomes from the analysis phase. If the validation activity is unsuccessful, the reasons are re-addressed in the development stage again. If the validation is successful, the test phase is complete and the data integration activity is ready for the deployment and release phase.

Figure 4.7 shows a UML Component Diagram representation of the test phase activities where validation checks and measures of the development phase activities are measured to progress to the deployment and release stage or return to the development stage for further development work.

![UML Component Diagram of Test Phase Activities](image)
The test phase validation involves examining the technical integration method and the flow of the data between data sources. It involves testing the components of the integration activity to ensure they work as intended and achieve the expected outcome such as the web application script, the target data sources, the data that is presented to the user from the data sources and the validity of this data (see sections 4.2.4 and 4.3).

Once the test phase is finalised and the tests are successful, the integration activity is ready for the deployment and release phase.

### 4.2.3.5 The deployment and release phase

The deployment and release phase is where the integration activity is ready to be made live and implemented. It has gone through the test phase successfully and is ready to initiate for real integration with live data sources.

The web applications integration is implemented in this stage with the target data sources and the data integration can now begin the final evaluation of the quality phase.

### 4.2.3.6 Quality phase

The quality phase is an evaluation phase to identify the usability, performance and accessibility of the integration activity. Figure 4.8 illustrates a UML Component Diagram of the workflow of the quality phase activities.
Figure 4.8: UML component diagram of the quality phase activities

Figure 4.8 shows a representation of the quality phase activities to determine the quality of the integration activity by examining its usability, performance and accessibility. The usability refers to a user perspective on the outcome of the integration between the web application and its data sources. The usability would be rated by the user as having suitable characteristics for their purpose or poor characteristics. The user evaluation is based on determining how usable the integration is and if the data transformation and presentation between data sources is accurate and stable.

Building from usability is the quality of the performance. The user rates the performance based on their experience during the integration where the speed of the data transformation and presentation between sources is measured along with the ability to handle unpredictable data and ongoing integration is proven successful. As discussed in McCarthy, (1998), Vosgien, (2015) and Richards, (2007), these are important quality measurements. The quality of the performance can be both its current performance for an integration activity and its ongoing performance into the future for different integration activities such as evolving data sources.

The accessibility refers to the uptime of the integration activity and the reliability to integrate with data sources due to availability of the integration service as discussed in Olsina, Papa, Molina, (2008). If the data sources can't be integrated with due to down
time, the accessibility measure fails. This is a yes/no approach and its implementation discussed in Section 4.3. Testing the continued accessibility of the integration with data sources is a core quality indicator as discussed in Alexander, (2014). This involves testing the components of the integration activity to ensure they work as intended and achieve the expected outcome such as the web application script, the target data sources, the data that is presented to the user from the data sources and the validity of this data.

4.2.4 Instrument

The methodology acts as a quantitative instrument enabling a data integrator to assess their integration goals to see where their web application sits and what actions to achieve their outcomes. The instrument acts as a procedure that can be followed by the data integrator. This approach is a structured approach allowing a data integrator to bring their broad integration goals and establish what the relationships are between the integration variables, apply the classifications scheme and the methodology.

4.2.5 Administration of the methodology

The methodology is instantiated with a support framework. The methodology acts as a discipline and set of procedures to follow as part of the integration process with the framework acting as a support tool to guide a data integrator in choosing an integration approach or technology for their integration activity. This framework is discussed in section 4.3 including an example of how the framework is applied. Detailed cases of how it supports the integration of data sources for different web applications are discussed in Chapter 5.

4.3 Framework

4.3.1 Overview

The proposed approach Generic Integration for Web Applications (GIWeb) is instantiated through a framework. It acts as a pluggable tool for the data integrator and
This framework is designed to be universal for integrating data sources for use in web applications. This approach addresses the difficulties associated with existing model, technical approaches and management process-based approaches which in turn enables a data integrator to avoid those same problems with their integration process by using the framework as their guide.

This section provides an introduction into the framework. The framework will be a development type framework that facilitates different technologies and acts as a support framework for a data integrator for their integration activities. It acts as a workflow for developing dynamically integrating applications and helps formalise and standardise procedures that a data integrator must follow for their data integration activity.

The framework acts as a support guide for a data integrator such as a web developer to consult with for their data integration activity. They can use this guide to evaluate their web applications characteristics to classify their web application determining which integration path to follow. A data integrator will be able to analyse the characteristics of their web application to see how it fits with similar web applications to then follow a recommended integration approach based on those characteristics.

Consider the following example. A web developer is creating a web application that needs to read data from three different external XML feeds. Using the support framework, they can examine the characteristics of the web application they are developing to best classify it in contrast to similar web applications. By applying the framework, the web developer proceeds through a self evaluation process where they evaluate the web application characteristics to find where it best fits for different integration approach options which then enables the data integrator to narrow down their integration technology approaches. It measures if an API is used or if it's a custom integration approach. The framework examines if the web application is programmable in nature and if data sources can be queried in real time. It evaluates if data sources require direct database access and if translation of data occurs. It also examines if the web application involves writing meta-data, how self-manageable it is and if it's centralised. The outcome of evaluating the web applications characteristics against these criteria narrows down the possible approach options such as models and
integration technologies to choose from for their integration goal. Section 4.3.2 discusses this structure and measures for evaluating the web application characteristics.

4.3.2 Structure

The framework is structured in a survey style with recommendations based on each outcome. The framework presents a series of questions based on determining the characteristics of a web application and expected integration outcome. These questions allow a data integrator such as a web application builder to contrast their web application against the framework criteria and classify it under a specific category of web application. Under each category are recommended approaches and technologies to use for that specific web application type and integration aim.

The framework enables a data integrator to utilise experience and lessons learned from other relevant integration projects to consider potential impacts for their integration activity. The framework is developed to reduce the number of options pruning non plausible integration approaches.

The framework is broken up into the following three phases:

**Phase 1: Identify characteristics of the web application**

**Phase 2: Compare web application characteristics to characteristics of suitable approaches**

**Phase 3: Identify appropriate technology to implement for integration process**

Figure 4.9 illustrates a tree diagram of the three phases of the framework starting with the initial criteria checks in the framework to narrow down the integration approach options which then enables the data integrator to narrow down their integration technology approaches. Table 4.2 discussed further examines how to narrow down and choose technology options.
Figure 4.9 is a representation of the support framework showing the initial criteria checks in the framework to narrow down the integration approach options. Further discussion in this chapter will examine different layers of the framework. The tree is used to construct a reduced number of alternative architectures. The number of alternative architectures that can be constructed from numerous options continues to be narrowed down by answering the initial classification questions. A systematic and disciplined approach to architecture definition is required to narrow down approach options and technologies to choose from.

The initial criteria checks from the first phase of the tree diagram are made up from the classification scheme dimensions as described in Section 2.3.3: data translation dimension, programmable dimension, centralised dimension, meta-data dimension and self-manageable dimension.
Based on answering the questions during the first phase of the framework, the data integrator is able to identify the type of approach they should be looking for. They can also rule out approaches that will not be appropriate for their type of integration process. The approach options may be based on characteristics such as being self-manageable, meta-data based and able to support real time querying which would suggest an XML base approach is most appropriate. Table 4.1 illustrates this example. It describes an example online repository of real time agriculture chemical prices web application, comparing its characteristics in response to framework criteria checks.

<table>
<thead>
<tr>
<th>Web Application Characteristic</th>
<th>Criteria Response</th>
<th>Narrowed Down Approaches</th>
<th>Narrowed Down Technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uses API</td>
<td>Yes</td>
<td>XML feed</td>
<td>RSS</td>
</tr>
<tr>
<td>Programmable</td>
<td>Yes</td>
<td>Schema mappings</td>
<td></td>
</tr>
<tr>
<td>Data sources queried in real time</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Translation</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Centralised</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Meta-data</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Self-manageable</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1 Example web application name: online agriculture chemical prices

In this example, the web application is categorised into different classification groups from the classification scheme. This enables the data integrator to narrow down the available approaches to use for their integration process. In this instance, after the criteria responses to the web application characteristics, they have narrowed down an XML feed and Schema mappings as suitable approaches. This then narrows down appropriate technologies to use for the integration process which in this example is an RRS feed.

In the first phase of the support framework, the criteria checks narrow down the characteristics of what integration approaches to consider for the integration process. The data integrator can then make a decision from a refined list of suitable data integration approaches to implement for their data integration process. This second
phase of the support framework for narrowing down approach types such as models and integration techniques is discussed below in Table 4.2. This table illustrates how the framework supports the data integrator by categorising the characteristics of different approach types and listing available suitable technologies based on these characteristics. These approaches are the specific technologies and approach types that are grouped under the approaches discussed in Chapter 2, section 2.4 and section 2.5.

The technology options can be determined once the approach options are narrowed down by progressing through the first phase of the tree criteria checks. The available technology options for a data integration process can be determined by progressing through a series of checks similar to the ones from phase one from the tree diagram (Figure 4.9). This later level of checks requires the data integrator to contrast the available technologies to choose from against a series of criteria checks. This enables them to scale down the available integration technologies to choose from for their integration process.

Figure 4.10 illustrates a tree diagram of how the data integrator can scale the available technologies to choose from in their data integration process. This is the final layer of the tree diagram Figure 4.9 for narrowing down technologies.

Figure 4.10: Tree diagram of available technologies

In Figure 4.10, the data integrator already knows the available approaches to choose from by completing the first phase of criteria checks from the tree diagram Figure 4.9. This stage asks the data integrator to compare if the available technologies incorporate the same data sources that the chosen approach requires. If the available technology doesn’t, it can be ruled out immediately as a suitable technology to consider for the integration process. The second check tests if the available technology meets the web application characteristics from the chosen approach. The final check tests if the technology fits the chosen approaches architecture. If each of these checks is answered
yes, the data integrator knows they can include this available technology as a viable choice to use for their integration process. If there is a no answer to any of these checks, the data integrator can rule out that particular technology as a suitable option for their integration process.

If an API is appropriate, the framework enables the testing of the different API characteristics to determine the appropriateness of an API. The API characteristics include:

- Testing what representation formats will be exposed
- Handle hypermedia linking (XML or JSON)
- How errors are reported back
- How request methods are handled that aren’t supported
- Which HTTP methods are available for each resource
- Handle features such as authentication (HTTP authentication, OAuth2, or API tokens)

The workflow is detailed in Figure 4.11
The final determination for choosing an appropriate technology to use in an integration process is to compare the different technology choices against the requirements characteristics described in Chapter 3. In a situation where there are multiple technology choices to consider after going through the tree workflow phases, the data integrator can further narrow down appropriate technology choices through another tree layer. This final tree layer draws on the requirements characteristics described in chapter 3 where the data integrator will analyse the various dimensions of an integration technology.

Figure 4.12 illustrates this layer of the tree and its work flow.
In figure 4.12, the data integrator assesses the technology approach against each of the criteria defined to rate it against these options. If one technology choice performs better than a counterpart, then it would be the more appropriate technology to implement for
the integration process. A technology approach doesn't have to rate well across the full spectrum of options to be the most appropriate technology choice. But when a data integrator is considering multiple technology options, the tree workflow from Figure 4.11 will help determine which technology has the more ideal characteristics to suit their integration process and therefore be the more appropriate choice.

Table 4.2 describes the combined layers from the tree workflow (Figure 4.9 and Figure 4.10) applied to a range of different web application types. Figure 4.9 illustrated a tree diagram of the three phases of the framework. Figure 4.10 illustrated a tree diagram of how the data integrator can scale the available technologies to choose from in their data integration process. Table 4.2 shows the narrowed down approach types categorized by web application characteristics and then available approach and technologies to use in the integration process for that approach. Each of the following describes an approach with a description, characteristics, its data sources and the available technologies for implementing the approach based on following this tree structure. Chapter 5 further discusses how these layers are used for a range of different web application types.

<table>
<thead>
<tr>
<th>Web Application Characteristics</th>
<th>Available Approach: XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uses API, Programmable, Data sources queried in real time, Has Translation, Meta-data</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Brief Description</th>
<th>Characteristics</th>
<th>Data Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Really simple syndication</td>
<td>Integrating web feeds using XML format. Based on parsing XML via a URL string.</td>
<td>Data Sources: SQL, XML</td>
</tr>
</tbody>
</table>

| Technologies for this approach: | RSS |

<table>
<thead>
<tr>
<th>Web Application Characteristics</th>
<th>Available Approach: Schema Mappings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uses API, Programmable, Data sources queried in real time, Has Translation, Meta-data</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Brief Description</th>
<th>Characteristics</th>
<th>Data Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mapping one schema set to another</td>
<td>XML based, meta-tagging based, SQL query based, programmable</td>
<td>Data Sources: SQL, XML</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------------------------------------------</td>
<td>-----------------------</td>
</tr>
</tbody>
</table>

**Technologies for this approach:** Schema mapping tools, XSD modeling toolkits

**Web Application Characteristics**

Uses API, Programmable, Has Translation

**Available Approach:** Middleware

<table>
<thead>
<tr>
<th>Brief Description</th>
<th>Characteristics</th>
<th>Data Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provides query language used to combine, contrast, analyze and manipulate the data</td>
<td>Combine data from multiple sources in a single SQL statement</td>
<td>Data Sources: SQL</td>
</tr>
</tbody>
</table>

**Technologies for this approach:** Open Grid Service Architecture

**Web Application Characteristics**

Uses API, Programmable, Data sources queried in real time, Has Translation, Meta-data

**Available Approach:** Meta-data model

<table>
<thead>
<tr>
<th>Brief Description</th>
<th>Characteristics</th>
<th>Data Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>No common schema for meta-data Defines Meta-data for the datasets Defines Meta-data for the Database system to enable querying and defining activities</td>
<td>SQL querying, meta-data and meta-tagging approach</td>
<td>Data Sources: SQL, Meta-data</td>
</tr>
<tr>
<td><strong>Technologies for this approach:</strong></td>
<td>Schema file</td>
<td></td>
</tr>
</tbody>
</table>

169
<table>
<thead>
<tr>
<th>Web Application Characteristics</th>
<th>Uses API, Programmable, Has Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available Approach: Component based framework</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Brief Description</th>
<th>Characteristics</th>
<th>Data Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Based on the common data model</td>
<td>SQL querying, meta-data and meta-tagging approach</td>
<td>Data Sources: SQL, Meta-data</td>
</tr>
</tbody>
</table>

**Technologies for this approach:** Common Model

<table>
<thead>
<tr>
<th>Web Application Characteristics</th>
<th>Programmable, Has Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available Approach: Brokers</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Brief Description</th>
<th>Characteristics</th>
<th>Data Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uses a software service approach to collect and integrate diverse data from autonomous data sources. The broker is able to collect the data from the source and determine how and where it goes towards its destination, acting as a negotiator between the source and the destination</td>
<td>SQL querying and programming approach</td>
<td>Data Sources: SQL, Meta-data</td>
</tr>
</tbody>
</table>

**Technologies for this approach:** Broker Tool

<table>
<thead>
<tr>
<th>Web Application Characteristics</th>
<th>Programmable, Has Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available Approach: Standardisations</td>
<td></td>
</tr>
</tbody>
</table>
The streamlining and consistency of the data, to standardise the data from one data source to another

<table>
<thead>
<tr>
<th>Available Approach:</th>
<th>Characteristics</th>
<th>Data Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standardisation</td>
<td>SQL querying, streaming,</td>
<td>Data Sources:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SQL, Meta-data</td>
</tr>
<tr>
<td>Technologies for this approach:</td>
<td>Standardisation tool</td>
<td></td>
</tr>
</tbody>
</table>

**Web Application Characteristics**

Uses API, Programmable, Data sources queried in real time, Has Translation, Meta-data, Self-Manageable

**Available Approach: APIs**

<table>
<thead>
<tr>
<th>Brief Description</th>
<th>Characteristics</th>
<th>Data Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application programming interface – a set of libraries and procedures on how to integrate applications with data sources</td>
<td>SQL querying and programming approach</td>
<td>Data Sources: SQL, Meta-data</td>
</tr>
<tr>
<td>Technologies for this approach:</td>
<td>API</td>
<td></td>
</tr>
</tbody>
</table>

**Web Application Characteristics**

Programmable, Has Translation

**Available Approach: Extract Transform Loads**

<table>
<thead>
<tr>
<th>Brief Description</th>
<th>Characteristics</th>
<th>Data Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extracting data from outside sources and transforming it and loading it into the target source such as a data warehouse</td>
<td>SQL querying, transform and load to target source</td>
<td>Data Sources: SQL</td>
</tr>
</tbody>
</table>
4.3.3 Applying the framework to any integration scenario

For a data integrator to implement this framework in any integration scenario, the data integrator simply follows the phases beginning with identifying their web applications characteristics through the classification phase described in section 4.3.2. The data integrator can then look for an available approach that is suited to those characteristics. From there, they can determine the technologies to implement for their integration process. Regardless of the number of approaches that could be considered, examining their characteristics will group the approach into the characteristics described in section 4.3.2. The data integrator can eliminate inappropriate approaches by applying the framework to their integration process. They can also categorise suitable approaches to consider using and the technologies that will be relevant to implement the chosen approach.

4.3.4 Framework supporting requirements

The framework supports the system and data integrator requirements described in section 3.5.3. For the system requirements, the framework supports the design process where the integration process describes the architecture being integrated with the goal of modeling the data integration landscape to be implemented. The data integrator is able to map out the right path to follow and providing a structured design process in their data integration process. The framework supports heterogeneous environments, integration from a variety of sources consisting of different structures and formats from mainframes to messaging systems. The framework supports structure and integrity during the integration process for ensuring data quality and data governance. The data integrator is able to choose a data integration process for maintaining optimal performance when integrating between a data source to its destination.

Table 4.2: Narrow down approach types

<table>
<thead>
<tr>
<th>Technologies for this approach:</th>
<th>Transform load</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Technologies for this approach:</th>
<th>Transform load</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technologies for this approach:</td>
<td>Transform load</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Technologies for this approach:</td>
<td>Transform load</td>
</tr>
</tbody>
</table>
The integrator requirements are met with the process of integrating data sources for the purposes of presenting the data in a web application must provide a means of accessing the data from the various sources in a consistent, timely and accurate manner. GIWeb also assists the ability to implement, manage and control the integration process without the ongoing reliance on a third party or highly skilled technical person.

4.4 Summary

This chapter proposed an integrated technology and process approach to improve on the current approaches for the integration of data sources for use in web applications. It proposed an approach to solving the problems discussed in chapter 2. It is known as Generic Integration for Web Applications (GIWeb).

It introduced a high-level methodology to support the data integration process for a broader set of integration goals. It is a methodology for data integration for web applications instantiated through a framework. This methodology supports the classification scheme presented in Chapter 2 and supports the requirements from Chapter 3. It acts as a high-level methodology that can be applied to all data integration scenarios.

GIWeb can be used to consult with and aid in attempting to integrate with data sources for web applications. This approach for data integration is a goal centric approach with a top-down design where data integration involves working out the goal first, identifying where the web application fits, applying a methodology and importing the technology to apply it.

Section 4.2 described the proposed methodology. It outlined its architecture and how it aims to improve on the current approaches discussed in Chapter 2. It also discussed how it will support the requirements presented in Chapter 3. This section detailed how a solution to the problems discussed in Chapter 2 would be to formalise and standardise procedures that a data integrator could follow for their data integration process. This would become a workflow for developing dynamically integrating applications through a
support framework. The methodology acts as an organised, documented set of procedures that can be used as a step by step approach to carrying out the intended integration procedure. It will also act as an objective set of criteria for determining whether the results of the procedure are of acceptable quality. It helps determine goals, motivation and purpose for the integration task and allows an integrator to assess where their application fits and what integration process to follow. It is used to consult and aid the integration of data sources through web applications. The methodology is divided into six phases: Analysis phase, Classification phase, Development phase, Test phase, Deployment and Release phase and Quality phase. Prior to these steps are pre-initiation activities including determining goals and motivation, problem definition and web application classification.

Section 4.3 describes how the methodology is instantiated with a support framework. The methodology acts as a discipline and set of procedures to follow as part of the integration process with the framework acting as a support tool to guide a data integrator in choosing an integration approach or technology for their integration activity.

This framework acts as a generic framework and guide that a data integrator could follow to achieve their integration goals. This approach enables the data integrator such as a web developer to choose the most appropriate integration approach to implement. The data integrator will be then able to choose an approach and data integration technology to use for their integration process. The framework is broken up into three phases: Identify characteristics of the web application, compare web application characteristics to characteristics of suitable approaches and Identify appropriate technology to implement for integration process. The framework supports the system and data integrator requirements described in section 3.5.3.

This chapter described the process for a data integrator to implement this framework to aid their data integration process by following the phases beginning with identifying their web applications characteristics through the classification phase described in section 4.3.2. The data integrator can then look for an available approach that is suited to those characteristics. From there, they can determine the technologies to implement for their integration process.
5 Applying GIWeb to case studies

5.1 Overview

Chapter 4 presented the design of GIWeb, an approach for supporting the integration of data sources for use in web applications. It consists of a methodology and supporting framework. This chapter illustrates how the proposed approach GIWeb is used in a variety of case studies. This chapter presents several cases where the approach was used to implement the integration of data sources for use in web applications in each case study.

The case studies chosen discuss a range of development scenarios that cover different types of integration environments. Section 5.2 explains the choice of a programming environment using the Zend platform and provides some background, while Section 5.3 describing the application of GIWeb to a Zend platform development scenario. Section 5.4 examines the application of the proposed approach to a project management framework PRINCE2 where the conceptual management of an integration project is modeled applying GIWeb in Section 5.5. Section 5.6 focuses on an integrated development environment using the Eclipse framework. The varying types of integration environments shows how the methodology and framework interacts with the whole development process and the benefits of it. Each case study has different expected outcomes. This includes generating technologies to implement in an integration process and generating code to connect to data sources and generate a process for the data integrator to follow. Section 5.7 presents a summary of this chapter.

5.2 Application to the Zend Framework

5.2.1 Overview

The Zend Framework is used to build websites (Zend, 2015). In this case, a web developer develops a web application built using the Zend framework due to its
suitability for ease of use and well documented online support. Alternatives such as Ruby on Rails, CakePHP will likely require more time to implement and have a steeper learning curve. To demonstrate a range of diverse development scenarios, a programming environment is appropriate given how web applications are developed. Zend is chosen in this work since it is a widely used web programming environment. This application uses a database and a web service as data sources. The aim is to structure the Zend model to support multiple data sources. Zend framework allows for various concepts that provide a solution to this scenario such as DataMapper Pattern, Service Pattern and Adapter Layer which are each software template patterns. However, this leads to confusion from the web developer as to how to put this all together into a reusable and scalable codebase.

5.2.2 Case study

It is important to understand an integration scenario that represents common website functions and activities. Ecommerce functions are a major component of the Internet ranging from online shopping stores, payment gateways for engaging services through to purchase upgrades on free to download mobile phone applications as discussed in Ceri, Brambilla, & Fraternali, (2009). A payment process is therefore an appropriate example to discuss. In this scenario, a web developer is developing a retail web application that consists of a series of web forms. The forms need to interact with an external payment gateway during the process and also cross-check some of the inputted fields such as the company registration number against an external company database to validate the information. When the process is completed there will be form information that is saved on a local database connection the web developer administers. The payment process is completed using an externally hosted payment gateway and some field information has been validated against using an external database. The web developer is using the Zend framework to facilitate the creation of the forms, modeling the development landscape and utilising the different Zend functions to connect all the sources together. It is an assumption for this case that the impact of choosing alternatives has been weighed up by the data integrator previously. Figure 5.1 illustrates a UML Component diagram of the retail web application. It shows the form validation
process with the external database and locally saving the form records to a MySQL database.

**Figure 5.1 UML component diagram of retail web application**

### 5.2.3 Problems and technology options

There are a number of potential problems associated with this case study. One or more of the data sources may require an application programming interface (API) to access them. Zend has solutions for this such as Apigility which is a program designed to overcome the difficulty with understanding how an API works and creating them. It does not require connecting programs to be built in the same supporting language that Apigility is native to which is PHP. Web developers can use any code they want. Apigility gives them a workflow to describe the API and an engine to run it on. They can take their code and plug it in to Apigility and deliver it as an API. Apigility takes care what’s required to set up an API, including the tasks of error handling, data validation, and setting up authentication.
Some data sources do not support APIs. In such situations, the web developer needs to find alternative methods to integrate them. The web developer may choose to specify all their workflows using XML. Each workflow consists of a setup section to support one or more data sources.

The data sources will be treated in completely different ways during the integration process. The payment gateway may involve using the Zend_Form_Wizard workflow for defining how the forms will interact with the payment gateways API. Apigility may also be an applicable technology here to streamline the creation of the API. The integration process for querying the business registration field, however, does not support an API and instead requires a specified data source connection established to query the information in real time. When using the Zend framework and without the use of GIWeb, the web developer in this instance has to undertake an enormous amount of work to reach their goals and go through some extremely involved and time-consuming development with lots of room for error.

5.3 Application of GIWeb

The application of GIWeb presented in Chapter 4 would be appropriate where a much more fine-grained control of the integration process is required by an application than is offered by Zend Framework. GIWeb is realised through a plug-in that features a wizard the data integrator follows to fine-tune their integration process. The GIWeb plug-in is an extension to the Zend platform that the web developer goes through prior to choosing the particular approach they are going to implement for their integration process. They are able to determine if the Zend Apigility approach is the best mechanism versus alternatives such as specifying the workflow using XML.

The application of GIWeb through a plug-in (see Section 4.2 and Section 4.3) allows the web developer to classify the web application characteristics. In this scenario the web developer needs to choose between an API for validating some of the form input fields using a Remote Procedure Call (RPC) and REST services, which are protocols to request a service from a program.
By applying GIWeb to test the applicability of Apigility, the web developer who is the data integrator in this scenario is able to test the appropriateness and readiness for using REST or RPC as an Integration approach with an API. The characteristics to determine this are tested as part of GIWeb acting as a support framework as illustrated in Figure 4.10. They include:

- Testing what representation formats will be exposed
- Handle hypermedia linking (XML or JSON)
- How errors are reported back
- How request methods are handled that aren’t supported
- Which HTTP methods are available for each resource
- Handle features such as authentication (HTTP authentication, OAuth2, or API tokens)

The GIWeb plug-in features a wizard which is a screen by screen process the data integrator goes through to determine these outcomes discussed. It is a pluggable module into the Zend framework. The data integrator progresses through the screens sequentially and by the end of it they will have narrowed down the addressing the criteria above.

The remainder of this section is as follows, section 5.3.1 presents a conceptual overview of the implementation. Section 5.3.2 describes how its build and includes how the GIWeb plug-in is merged with the Zend Framework, 5.3.3 discusses how it is used and section 5.3.4 reviews what the implications are for this approach.

**5.3.1 Conceptual overview**

This section presents a conceptual overview of the GIWeb plug-in that a data integrator can follow to fine-tune their integration process when using the Zend Framework. It describes the process the data integrator goes through when using Zend’s Apigility for implementing the GIWeb plug-in for their integration process.
The data integrator uses APILgility as part of the Zend Framework to integrate their web application with data sources using an API. The workflow describing how Zend facilitates the integration through the API is represented in Figure 5.2. This shows the workflow prior to implementing GIWeb. The data sources consist of various services and data from an API provider. An API provider is a core stakeholder who owns the data or services such as a business that creates the data. The API publisher is a component that publishes the API by consuming the services and data using an API solution such as a web application designed for the specific purpose of establishing a connection with the services and data. The API developer is what the data integrator is, a web application builder who uses the API for their solution. When developing their web application and using the API to integrate, they use solutions under the Zend Framework such as APILgility to develop. The web applications are the product they have developed leading to the end users who are the application consumers.

![Diagram of API workflow]

Figure 5.2 Zend framework in the API workflow

The conceptual overview of the GIWeb plug-in that a data integrator can follow starts with APILgility dashboard as shown in Figure 5.3. This is enhanced with the GIWeb plug-in so that when a data integrator creates their API through the APILgility wizard, the GIWeb plug-in described in this section first helps them determine the most appropriate API approach to use.
Once the GIWeb plug-in has been installed into APIgility, the data integrator is presented with a wizard to follow when they click to create a new API as shown in Figure 5.4.
The current options presented when clicking create API are for the data integrator to choose between REST Services or RPC Services as shown in Figure 5.5. The wizard changes this process so that the data integrator first goes through the wizard and its process to filter which of the two service options is the most appropriate to choose.
The wizard starts with defining the web application as shown in Figure 5.6. This includes selecting the API type and deciding if the web application is programmable or not, if it will involve real time querying, translation, if it involves using meta-data and if it is self-manageable.
### Define the Application

<table>
<thead>
<tr>
<th>Web Application Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>API type:</td>
</tr>
<tr>
<td>Programmable:</td>
</tr>
<tr>
<td>Real-time querying:</td>
</tr>
<tr>
<td>Translation type:</td>
</tr>
<tr>
<td>Meta-data type:</td>
</tr>
<tr>
<td>Self manageable</td>
</tr>
</tbody>
</table>

Figure 5.6 Define the web application
Figure 5.7 shows defining the data sources using the wizard.

Figure 5.7 Define the data sources

Figure 5.8 shows the final screen of the wizard is the classification component to further classify the web application to best determine the most appropriate approach.
Figure 5.8 Classification

The result is that the most appropriate API type will be known and presented to the user to define the rest of their integration process as shown in Figure 5.9. In this scenario, REST services is the most appropriate type based on answering the prompts from the wizard. The web developer also has categorised and determined supporting information for their integration process to validate data source integration methods that the API will use.
5.3.2 Implementing the GIWeb plug-in

This section describes how the GIWeb plug-in is merged with the Zend Framework. APIgility has an editable configuration file which enables further extensions to be included. The GIWeb plug-in discussed in this section can be included as an extension by editing the configuration file to include it. The configuration file is included within a web application by including a single line of PHP code:

```
include "config/application.config.php";
```
In this instance, this statement is written in PHP but similar methods would be applicable for other programming languages.

5.3.3 Implementation

This section presents how the configuration file looks after the changes described in section 5.3.2 have been implemented. The highlighted yellow section is the additional include for the GIWeb plug-in.

See Appendix 1

5.3.4 Implications

This section describes the implications for a data integrator when using the GIWeb plug-in within the Zend framework in an integration process.

For a first time user to the Zend framework using APIgility, there may be a technical learning curve in setting up the plugin. The consequences of choosing the wrong approach or technology could result in the data integration failing or only achieving a partial success, poor quality data results or incorrectly matched data fields (see Section 2.5). Therefore, the extra time planning and setting up the plugin is well worth the effort. The plugin requires technical knowledge of how to use the APIgility and an understanding of configuring the source code which is written in PHP. The plugin involves extra thought and planning for the data integrator during their integration process with data sources. It is an additional step to their development process when using APIgility which requires additional time to progress through and implement via the plugins wizard. The implication of this extra step is additional time commitment to this part of the integration process in planning and setting up the plugin which will need to be prepared and allocated for in the project plan.
5.4 Application to the PRINCE2 framework

5.4.1 Overview

PRINCE2 is a software management framework. It is a methodology that encompasses the management, control and organisation of a data integration project (Richards, 2007). It is a process driven and structured project management approach describing the procedures to coordinate the people and activities of a project, how to design and coordinate the project (Richards, 2007).

5.4.2 Scenario

A project manager is coordinating the implementation of a web application into their medium sized organisation. They have adopted the PRINCE2 framework as the model to follow for governing all aspects of the projects implementation. The web application is an incident management system designed to be used by all staff across the organisation for reporting incidents. The system needs to pull data from internal and external data sources. The project manager is coordinating how these sources will be pulled together for the web developer to access to achieve the data integration stage of the project workflow. Internal sources include an existing Excel spreadsheet containing old incidents that were recorded in a manual, paper based approach. It also includes an old Access database which recorded some aspects of an incident on a standalone database. External sources include a cloud based staff management system where records such as contractors, insurance information and a different cloud based system that stores all the staff records. Figure 5.10 illustrates a UML Component diagram of the different data sources that a staff member needs to utilize together in the incident management system.
5.4.3 Problems and technology options

In this scenario, the project manager is acting as a data integrator by determining how the different data sources are to be integrated with through the PRINCE2 workflow. But they face several challenges and need to address which technologies to select for the integration process.

PRINCE2 is considered inappropriate where project requirements are expected to change rapidly (Richards, 2007). Data integration projects deal with unpredictable data and as the project evolves, the requirements will change rapidly. This project needs to take into account the maturity of multiple external sources that each may evolve and expand in data and fields at different rates. The cloud based contractor system may rapidly evolve in additional fields and changes separate to the way the staff management system changes. The internal sources described including the Excel spreadsheet and the Access database are static and unlikely to change since these are controlled by the organisation themselves.
Critics of PRINCE2 projects (Richards, 2007) suggest that high maintenance requirements could hinder the speed of delivery and the ability to adapt to change for this project. Documentation implemented in an overly bureaucratic way described as a common mistake (Richards, 2007). PRINCE2 also requires a high-level of training for team members in the project according to the Office of Government Commerce. (2002).

Given there is a wide range of available technologies to implement for the integration process, deciding on one presents several challenges. The decision making stage involves strong reliance on team member planning for technology selection. This method is not appropriate as the outcomes of a chosen integration technology are going to be tested against and known fully at the planning stage. Without the use of GIWeb, there is room for error and choosing the wrong approach at the planning stage could have drastic effects further down the project during its implementation or evaluation stages.

For example, in the case discussed in this section, the technical approaches considered include using an API for the integration process with the cloud based contractor system compared to an XML feed. Both options have different approaches for the implementation and expected integration outcomes. The decision to choose one over the other is based on a general consensus of the PRINCE2 project team. There is at no stage a selection evaluation process to test the appropriateness of one technology approach over another.

5.5 Application of GIWeb

The application of GIWeb proposed in Chapter 4 is appropriate to create a wider selection of opportunities for the project manager in their PRINCE2 integration process. GIWeb can be realised through a plug-in that a web developer can follow to fine tune the integration component in the PRINCE2 workflow. A model based process helps to better understand an integration scenario that represents a different end of the development spectrum to a programming environment. PRINCE2 is chosen as it is a widely used project management model and is more suitable for web application projects than alternatives such as PMI from the Project Management Institute or APM from the association of project management. Project management forms a major part of planning
and delivering web applications in the commercial sector and it is appropriate to use a common and widely used model as an example case due to its use in the field.

The project manager will add the GIWeb plug-in to the data integration component of the PRINCE2 workflow. The web developer who follows the workflow in the PRINCE2 model will use the GIWeb plug-in to improve their selection of the most appropriate integration technology for the integration process. This will be done through following a wizard as an action the web developer will go through prior to choosing the particular approach they are going to implement for their integration process. They will be able to determine if an API or XML is the best integration method.

Section 5.5.1 presents a conceptual overview of the implementation. Section 5.5.2 describes how its build, 5.5.3 discusses how it is used and section reviews what the implications are for this approach.

5.5.1 Conceptual overview

This section presents a conceptual overview of the wizard that a data integrator can follow to fine tune their integration process when using PRINCE2. It describes the process the data integrator goes through when using PRINCE2 for implementing the wizard for their integration process as part of the PRINCE2 modeling process.

Figure 5.10 shows the PRINCE2 model and where the framework fits in this model. The framework is part of the work package component of the PRINCE2 framework where the Project Manager in the controlling stage interacts with the framework to determine the appropriate integration process to follow. This is a process prior to implementing the work packages. It leads to justification in the decisions made when implementing the work package for the Integration process. By following the proposed frameworks workflow, a validity check has been performed to justify the integration process chosen. The result enabling them to issue the appropriate integration process work packages and when completed, an integration plan is one of the MP outputs.
5.5.2 Implementing the GIWeb plug-in

This section describes how the GIWeb plug-in is merged with PRINCE2. The PRINCE2 model is a conceptual one that a project manager follows and implements for their project. The GIWeb plug-in is also conceptual acting as a work flow extension. It is an addition to the process stage of the PRINCE2 framework. This stage describes progressive steps through the project lifecycle, from getting started, delivery tasks and outcomes to responsibilities, project approval and closure (PRINCE2, 2014).

The integration framework sits in the process stage as a new activity interacting with the controlling stage. The integration task is issued from the controlling stage (CS), implemented and the result returned to the controlling stage. From there the rest of the
PRINCE2 workflow continues progressing to the project delivery being accepted and project management outputs. The integration plan becomes a new extension to the project management outputs.

5.5.3 Implementation

A number of project management software packages support the PRINCE2 framework such as CorePM, Workamajig, Wrike and P2ware. Many of these examples specialise in project management capabilities beyond PRINCE2 so for the purposes of this case, CorePM is chosen because it is specific to PRINCE2 and highly functional for this purpose. CorePM is a cloud based software package that is built on the principles of PRINCE2. It is an assumption for this case that the impact of choosing alternatives has been weighed up by the data integrator previously and they have considered the impacts of choosing the CorePM over alternatives. These impacts include choosing alternative project management software that isn’t as functional or easy to use when compared to CorePM. In the event an alternative was chosen that wasn’t as functional or easy to use, this would limit the ability to implement the PRINCE2 project in full and successfully. The impact of choosing CorePM itself is also reliant of the data integrator implement the PRINCE2 model correctly which is based on their own understanding and familiarity with the tool. It is an assumption for this case that the data integrator is familiar with CorePM in order to implement a PRINCE2 project model. When a project manager uses CorePM to model a project, they are structuring the work flow in line with the PRINCE2 methodology. Extending CorePM with the GIWeb plug-in discussed in this section would involve including the GIWeb plug-in as part of the process creation stage through CorePM. When included, the GIWeb plug-in becomes a task to follow when the part of the work flow it belongs to is executed.

In the example of using the CorePM product, the workflow is extended by including the activities of the GIWeb plug-on as a new process in the implementation stage. Figure 5.12 illustrates this example where the GIWeb plug-in activities have been added to the PRINCE2 workflow. In this example, a process is titled Integration with external cloud based systems describing the activity. In the scenario described in this section, two different cloud based systems are to be integrated with. The process will be the same for both integrations. The first implementation stage is to follow the integration framework to
determine the most appropriate approach for a data integrator to follow to achieve integration with the cloud based system. This involves following the GIWeb tree presented in Section 4.2.2. and Section 4.3.2. The follow on stage is to then determine the outcome of the initial criteria checks. This then leads to a narrowed down list of approach options and a decision on which approach is the most appropriate one to use. The final stage is to implement that chosen integration approach.

A process in PRINCE2 is managing product delivery (MP). A quality MP output is what is determined which is labelled as approach options as shown in Figure 5.12. This is an outcome from being narrowed down through the GIWeb criteria checks. This output effectively measures and ensures the quality of the integrity of the chosen data integration technology because the process has ensured a proper analysis of options and measuring them against the GIWeb criteria checks. Regardless of which technologies the web developer implements during the data integration stage of the project, the project manager has enhanced the process through a validation and verification workflow in choosing which technology to implement. Compared to not following GIWeb where there could be room for error in choosing the wrong integration
technology or a complete failure of that part of the project. This could result in increased costs, inaccurate results from the integration process, time delays or ongoing reliance on an unnecessary third party or external IT resource.

5.5.4 Implications

This section discusses the implementations the plugin to the PRINCE2 framework. The application of the GIWeb plug-in to a PRINCE2 project management software package such as CorePM involves additional preparation and planning time. The impact of not applying GIWeb in this case is that the project manager has a reduced quality output in the MP Outputs and therefore an increased risk in the project. There is risk of an incorrect integration technology being chosen by the developer because they have not gone through an evaluation of the technical outcomes examining the data quality potential of each integration option. The project manager will go through the workflow discussed in section 5.5.1 which is an additional activity to their other project management requiring specific preparation within the program and time commitment to implement. This extra planning and implementation for the project manager requires more time commitment. But the value added benefits of going through this process enables the project manager to validate and justify the decision for a particular integration process as part of their overall PRINCE2 project model. A discussion around the risks and impacts in relation to integration selection options and decision making is discussed in sections 6.4.3.1 and 6.4.3.2.

5.6 Application to the Eclipse platform

5.6.1 Overview

Eclipse is an integrated development environment (IDE) that can be used to develop web applications. This is a technology a web develop users to develop their web application and create the connections from their source code to target data sources. Eclipse is a programming based framework.
5.6.2 Scenario

A web developer is using Eclipse to develop a web application that is developed in PHP. The web application is going to be part of a mobile App on the iPhone where the PHP web based component is loaded from within the App. The web application will load new product price quotes for a user who logs into the App. When a new quote is ready, the web application also pushes a notification back to the user’s iPhone to let them know to log into the app and view their quote. Login details are stored on a MySQL database on the same server the PHP application resides. Quotes are stored on a different server managed by the supplier who issues the quotes. Their server runs a Cache database. Push notifications need to run through an APNS server such as Urbin Airship. Figure 5.13 illustrates a UML Component diagram of the quoting system showing the user login and the external source of the quotes.

![Figure 5.13 UML component diagram of mobile quoting app workflow](image)

5.6.3 Problems and technology options

Eclipse is used to develop the web application with PHP being the chosen language for the development. The web developer needs to achieve several goals.
- Establish a connection to the MySQL database where the login details for the App will be stored.
- Establish a connection to the Cache database where the quote information is populated.
- Connect to APNS server to pass push notifications

Each of these goals requires a different type of code connection to query and interact with each data source.

Each of these connections will require a different type of data source connection code. The MySQL database source will require generating connection code for a local query source. The Cache database source will require generating connection code for a remote query source. The APNS source will require generating code for a web service based source.

The Eclipse platform currently enables the web developer to go through a database selection process involving selecting the database type from a drop down, defining the user id and password, specifying a driver and database URL as detailed in Figure 5.14 below.
Figure 5.14 Eclipse database selection

The database types specified in the drop down in Figure 5.14 are static and do not reflect two of the data source types in this scenario, an external cache database and the APNS web service. Eclipse has been described as having difficulties with the database selection process. Without the use of GIWeb, the database selection process is not intuitive for driver and connection profile definitions (Graham & Payton, 2014). It has also been described as having not all data types correct and not all database objects appear in the DSE (Graham & Payton, 2014).

5.6.4 Application of GIWeb

The application of GIWeb proposed in Chapter 4 is appropriate to create a wider selection of data sources and assist in connecting to these data sources for the data
integrator in their integration process. GIWeb is realised through a plug-in that a web developer follows to fine tune the available data sources and improving the connection to these data sources.

The web developer will add the GIWeb plug-in to the Eclipse platform. Once installed, the web developer will use the wizard to improve their selection of the most appropriate integration data sources for the integration process. The GIWeb plug-in features a wizard that is an action that the web developer will go through as part of choosing the particular data source they are going to implement with for their integration process.

5.6.5 Conceptual overview

This section presents a conceptual overview of the GIWeb plug-in that a data integrator can follow to fine tune their integration process when using the Eclipse Framework. It describes the process the data integrator goes through when using Eclipse for developing a web application and implementing the GIWeb plug-in for their integration process. The GIWeb plug-in enables the data integrator to enhance their range of data sources to select from. The process describes the technical activities the data integrator implements to load the fine tuned range of data sources that were determined from following the GIWeb plug-in. Section 5.6.6 describes how the GIWeb plug-in is merged with the Eclipse Framework highlighting what has changed in the Eclipse configuration and Appendix 2 details the specific source configuration. An analysis of the impacts and results of extending Eclipse with GIWeb is presented in section 6.4.3.

The Eclipse Platform is built on a mechanism for discovering, integrating, and running modules called plug-ins (Eclipse, 2014). It principal role is to provide tool providers with mechanisms to use and rules to follow that lead to seamlessly integrated tools (Eclipse, 2014). Figure 5.15 shows the major components and APIs of the Eclipse Platform and the relationship to tool extensions. It consists of WorkBench, Jface, Standard Widget Toolkit (SWT), Workspace, and the Team and Help interface. In the scenario discussed in this section, the GIWeb plug-in focuses on the Workbench component.
The aim of the GIWeb plug-in is to enable the web developer to compare the criteria checks presented in Section 4.2.2 and Section 4.3.2 to narrow down the suitable data sources for their web application. The process of following the plug-in helps the data integrator generate a fine tuned list range of appropriate data sources for their integration process. It aims to enable the data integrator to load up a narrowed down list of appropriate data sources as an extension to the default data sources on the Eclipse framework. This acts as an extension to the Eclipse configuration and ensures the web developer has chosen the appropriate data source option for their integration process.

To achieve this in Eclipse, the web developer will go through the several steps to follow the GIWeb plug-in.

- The web developer uses Eclipse’s update manager to install the GIWeb plug-in
• In the "Select a wizard" dialog, select "Integration > mapping file" and click "Next"

• In "New Mapping Data File" dialog, select a "container" and a "file name" and click "Finish".

In the "Mapping Editor", Edit the "Connection info" and click the "Initialize entities from tables" button as shown in Figure 5.16.

![Figure 5.16 initialize entities from tables](image)

The web developer will modify the "Base name" of the class and "Field name" of field name.

"AttachFile" for example at the "Field type", will represent an attach file's name and generate source files to persist the attach file.

The web developer edits the "Output files dir", "Project package" and then clicks the "Generate source files" button as shown in Figure 5.17.

![Figure 5.17 Generate source files](image)

The .xml file generated downloads the dependent libraries

This approach assists the web developer by in generating connection specific code for different data sources. Once the GIWeb plug-in is loaded, the data integrator will be able to select from the relevant connection profile as part of the data source connection wizard in Eclipse as shown in Figure 5.18.
The data integrator can now change the source files to narrow down database selections. Using the GIWeb plug-in, the data integrator can establish a connection to the MySQL database, the Cache database and select a connection to the APNS server to pass push notifications.

### 5.6.6 Implementing the GIWeb plug-in

This section describes how the GIWeb plug-in is merged with the Eclipse Framework. Eclipse has a plug-in manifest file, plugin.xml, which defines how the GIWeb plug-in extends the Eclipse platform and is parsed when the plug-in is loaded into Eclipse. The extension involves extending the Eclipse workbench. The user interface plug-in is extended by the workbench plug-in via an actionSets extension that defines specific workflow related items for the data integrator to progress through. These items are
presented as a wizard that enable the data integrator to compare the criteria checks of the methodology and framework presented in Section 4.2.2. and Section 4.3.2.

The host plug-in is the Eclipse workbench user interface (UI) that can be extended via an extension-point known as actionSets. The UI plug-in uses the actionSets extension-point to extend the workbench UI plug-in with specific new menu items, in this case: Integration > Approach Criteria Checks. Figure 5.19 shows the relationship between these plug-in components, the extension Integration plug-in and the classes of the extension's callback objects.

![Figure 5.19 Plugin components](image)

The data integrator will go through the following steps to edit the plugin.xml file to define the integration wizard extension.

1. Right click on the plugin.xml or MANIFEST.MF in the project view.
2. Pick "PDE Tools->Externalize Strings".

3. Define strings to be moved and their property names in the plugin.xml file.

For the wizard, this would look like the below xml:

```xml
<property
    defaultDisplayName="Connection Properties"
    name="connectionProperties"
    canInherit="true"
    defaultDisplayName="integration.catalog"
    name="Integration_Catalog_File"
    type="string"/>
</property>
```

This is then paired with an entry in the .properties file found in:

```
"/myProject/OSGI-INF/l10n/bundle.properties"
```

New Entry: `org.eclipse.datatools.connectivity.oda.dataSource` extension point

It is also in the plugin.xml host file that the extension-point is declared in an extension-point XML element.

The implementation can be viewed in Appendix 2 detailing the source configuration.

### 5.6.7 Implications

This section discusses the implementations of application of the GIWeb plugin to the Eclipse Platform. Choosing an appropriate data source is an important step requiring
deep knowledge of many possible data source connections. The consequences of choosing the wrong data source could result in the data integration failing or only achieving a partial success, poor quality data results or incorrectly matched data fields (see Section 2.5). The plug-in requires deep technical knowledge of how to use the Eclipse framework and an understanding of database connections.

The GIWeb plug-in involves extra thought and planning for the data integrator during their integration process with data sources. It is an additional step to their development process when using Eclipse which requires additional time to prepare and implement via the plug-ins. The alternative to applying GIWeb in this case is that the data integrator would be forced to a trial and error in applying different integration technologies to each of the integration processes discussed. The increased number of attempts and potential errors could be very time consuming for the data integrator. By committing to the implementation of one technology, they won’t know that it is successful until the end of the integration process. Additional planning and time commitment is required to implement the GIWeb plug-in but the tradeoff is a reduction in the number of decisions required during the integration process.

The GIWeb plug-in requires extra coding to configure the Eclipse framework plugin.xml script with the changes discussed in Section 5.4.4.2 and Section 5.4.4.3. This involves writing extra lines of code on the plugin.xml file. A discussion around the risks and impacts in relation to integration selection options and decision making is discussed in sections 6.4.3.1 and 6.4.3.2.
5.7 Using GIWeb to develop an application

5.7.1 Overview

This section presents the development of a web application using a specific development environment and the application of GIWeb to this development process. It demonstrates the use of the GIWeb plug-in for developing a web application using the Zend framework. The application development used as a case study in this chapter is a customer relationship management system (CRM). A CRM is chosen since it is a widely used commercial example, organisations all over the world use a CRM to manage and engage their customers. It is a web based software tool for managing customer records, including tracking all aspects of the relationship with each customer such as conversations, enquiry or sale status and contact details. GIWeb is demonstrated to be viable for aiding the web developer in achieving their integration outcome. It illustrates how the framework is used.

The reminder of the section is as follows, section 5.7.2 describes the CRM system application and the data sources it uses. It presents the requirements to be addressed by a development solution. Section 5.7.3 discusses the application of GIWeb to a data integration process. Section 5.7.4 describes the implementation of GIWeb as a plug-in to the customer management system. It shows how the web developer follows the GIWeb plug-in and its outcomes. Section 5.7.5 presents the final configuration showing the integration outcome from applying GIWeb. Section 5.7.6 presents a summary of this chapter.

5.7.2 Case study: web-based customer management system

The customer relationship management (CRM) system discussed in this case is a web application that is used as a management tool for businesses to help manage customer relations. This involves collecting basic information about a customer which is then stored in a database. This includes the customer name, contact details and company
The CRM system sends the customer records to an external system that also needs this customer data. The web application is a PHP script developed using the Zend framework (described in Chapter 5) that queries a MySQL database. It consists of a web-based HTML form where a user enters their data. On submission, the browser forwards data to a server side PHP script that receives the form fields and inserts the records into the MySQL database. Figure 6.1 illustrates a UML Component diagram of the CRM workflow showing the form submissions of customer records, processing via a PHP script, storing in the MySQL database and integration via a web service to an external system.

![UML Component Diagram](image)

**Figure 5.20 UML component diagram of the CRM system workflow**

The integration challenge is to integrate some of these fields with an external server in real time and in a preferred format. The preferred format is an XML file that is read by the external server application. This is required locally on its end due to limitations with the ability to read external files. The external server has security and firewall restrictions affecting its ability to be able to establish a real time data source connection with the target web application. Instead, it will look at the XML feed and read from that. To address the integration challenge discussed, the web developer decides on which integration path to take for their integration process. The process is complicated by the
expected format of the files to be integrated. The format of the XML file needs to be very specific in structure and output to be interpreted by the external server.

The difficulty for the web developer is determining which technologies to implement (see Section 2.5). The commitment to choosing one particular technology to achieve the integration outcomes relies on deep knowledge of the different options to choose from and can result in the wrong process being implemented. The risks of improper data integration are well understood in enterprise systems development (Azadeh, Saberi, Ghaderi, Gitiforouz, & Ebrahimipour, 2008). By choosing the wrong integration technology, the web developer may spend time trying to implement something that isn’t going to work or isn’t going to produce the correct or optimal results. The result is an increased cost or total cost blowout to the project, a large amount of wasted time or even a total failure in the project outcomes. The web developer then has to start over and try a different technology (Laudon, & Laudon, 2004; Mork, Halevy, & Tarczy-Hornoch 2001). This decision making process is where GIWeb aids the web developer. A discussion around the risks and impacts in relation to integration selection options and decision making is discussed in sections 6.4.3.1 and 6.4.3.2.

Following the format for requirements presented in Section 3.5, Table 5.1 presents the Integrator requirements for a data integrator user such as a web developer.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>The process of integrating data sources in real time and in a preferred format.</td>
</tr>
<tr>
<td>2:</td>
<td>The ability to determine which integration technologies to implement</td>
</tr>
</tbody>
</table>

Table 5.1 Integrator Requirements
Table 5.2 presents the system requirements consisting of the technical outcomes that need to be supported.

- **1.1: Design Process**

  The integration process must describe the architecture being integrated with the goal of modeling the data integration pathway to be implemented. This will aid data integrators in mapping out the right path to follow and providing a structured design process for the data integration process. As part of identifying and modeling the integration landscape, validating the technical requirements for the implementation of the integration process is also integral.

- **1.2: Heterogeneity**

  The integration process must support heterogeneous environments, integration from a variety of sources consisting of different structures and formats from mainframes to messaging systems. This could include different file formats, access protocols and query languages, different ways of representing and storing the same data and even relationships between data.

- **1.3: Structure and Integrity Support**

  The integration process must support data quality and data governance. The data translated or transitioned from one source to another must not lose its integrity. Its format may change from one source to the other as part of the integration process but the information contained should be presented the same without any loss or misinterpretation from what as in the original source.

- **1.4: Performance**

  The integration process must ensure the data integration process maintains optimal performance when integrating between a data source to its destination.
There should be reduced lag in presentation or reduced loss in quality of data. The framework should have the ability to recover from errors such as source or target systems going down and data quality issues.

Table 5.2 System requirements

5.7.3 Use of GIWeb

GIWeb is applied to the Zend Framework that the web developer is using to develop the web application. This is realised as a plug-in. The GIWeb plug-in is an extension to the web development process that the web developer goes through prior to choosing the particular approach they are going to implement for their integration process.

The application of GIWeb to this case will help guide the web developer in choosing the most appropriate data integration technology to use for their integration process. GIWeb will help them identify characteristics of the web application, compare these characteristics to characteristics of suitable data integration technologies and ultimately identify the appropriate technology to implement for their integration process.

Figure 5.21 shows a UML Component Diagram of the integration process involving the web developer deciding on which technology approach is the most appropriate to implement. In the scenario discussed, the web developer needs to choose between an API for validating some of the form input fields using a Remote Procedure Call (RPC) or a direct XML based integration.
Figure 5.21 UML component diagram of the use of GIWeb

Figure 5.21 shows that using GIWeb, the web developer is able to test the appropriateness and readiness for each specific integration approach and the technologies needed to implement it. The web developer is able to determine what integration technology is the best mechanism versus alternatives for their integration activity. GIWeb allows the web developer to classify the web application characteristics and thereby narrow down the suitable technology options to choose the most suitable one for their integration process.

In the first phase of applying GIWeb, the criteria checks narrow down the characteristics of what integration approaches to consider. The web developer can then make a decision from a refined list of suitable data integration approaches to. The technology options can be determined once the approach options are narrowed down by progressing through the first phase of the criteria checks. This later level of checks requires the web developer to contrast the available technologies to choose from against a series of criteria checks. This enables them to scale down the available integration technologies to choose from for their integration process.
The data integrator can eliminate inappropriate approaches by applying the framework to their integration process. They can also categorise suitable approaches to consider using and the technologies that will be relevant to implement the chosen approach. Section 5.7.4 will discuss how GIWeb is implemented as a plug-in for the CRM system data integration process.

5.7.4 Implementing the GIWeb plug-in

The development of the CRM system follows the web development process as described in Section 4.2.1. The data collection form is a HTML to post the form fields to a PHP script that receives the data and inserts the records into a MySQL database (see appendix 3). In this scenario, the web development process is extended to include the web developer consulting the framework tree presented in Section 4.3.2 through a plug-in. By following this framework tree through the GIWeb plug-in, the web developer is able to classify the web application against the criteria checks.

The GIWeb plug-in presents a series of criteria checks for the web developer to contrast the characteristics of the web application against.

The GIWeb plug-in follows the phases described in Chapter 4:

Phase 1: Identify characteristics of the web application

Phase 2: Compare web application characteristics to characteristics of suitable approaches

Phase 3: Identify appropriate technology to implement for integration process

The web developer has a number of integration technology options to consider including using an API, message queueing, program-to-program communication, direct XML and JSON. In the scenario of Section 5.7.2, the web developer will progress through the GIWeb plug-in beginning with Phase 1. This phase examines the characteristics of the CRM system for using an API or if it is a custom built integration.
Figure 5.22 presents the first phase of examining the characteristics of the CRM system using GIWeb.

The CRM system discussed is custom built and the integration is not based around an API. The CRM system is programmable and its data sources can be queried in real time. But the integration component of the CRM system integration is with an external server. This external server cannot be queried in real time. It does involve writing some meta-data for matching and describing data fields. The CRM system is considered centralised and it does not require writing meta-data. Since the web developer is creating the CRM themselves, it is self-manageable.

The final determination for choosing an appropriate technology to use in the integration process is to use GIWeb to compare the characteristics of the different technology choices. The web developer assess the each technology against each of the criteria...
defined to rate it against these options. If one technology choice performs better than a counterpart, then it would be the more appropriate technology to implement for the integration process. Figure 5.23 shows the GIWeb criteria to compare.

Figure 5.23 GIWeb criteria for technology characteristics comparison
When applying the tree to the different technology options API, message queueing, program-to-program communication, direct XML and JSON, the resulting outcome is a narrowed down set of possible integration technologies that will be appropriate to choose from. In this case, these options have been narrowed down from five to two consisting of Remote Procedure Call (RPC) and a direct XML based integration incorporating the same data source as shown in Figure 5.24. Only RPC and direct XML were sufficient for connectivity, architecture and ease of use in this case given the specific formatting required and the required architecture and limitations with the external server.

![Technology Options Table]

Figure 5.24 Narrowing down suitable technology options for the data integration activity.

To narrow down which of the two technologies are most appropriate, Figure 5.25 shows how phase 2 of GIWeb examines if the integration technology options incorporate the same data sources that the CRM system requires based on the system and integrator requirements presented in Table 5.1 and Table 5.2.
Figure 5.25 GIWeb phase 2 characteristics

Following the GIWeb assessment of the web application characteristics reveals that both technology integration options meet the web application characteristics and fit the available approach architecture. HTTP methods are available, representation formats are acceptable but neither option results in errors reported back in an acceptable format. Authentication is supported in both options.

A direct XML based integration is the most appropriate integration technology for the web developer to follow. The web developer builds a custom FTP connection to the remote server that will populate it with a generated XML file. The alternative technology option of a RPC is ruled out. This is because the RPC option replies on the external server being queried in real time. This option is ruled out due to this characteristic being unachievable in the integration scenario discussed. The direct reading of an XML file
was also not applicable due to the external servers limitations. An external third party API was not appropriate due to the loss of control and flexibility in controlling how the integrating will work for the specific scenario. A custom API enables the web developer to have full control over the integration process to customise it for the specific requirements. Custom building a direct FTP connection to the remote server populating it with the generated XML file becomes the most viable option.

By not applying GIWeb, the alternative is that the web developer will face a trial and error scenario in applying different integration technologies to the integration process. Trial and error has been analysed in the academic community and was determined as important in cases like this. Mork, Halevy, & Tarczy-Hornoch (2001) has analysed it in the case of a model for data integration systems of biomedical data applied to online genetic databases. Azadeh, Saberi, Ghaderi, Gitiforouz, & Ebrahimipour, (2008) has analysed it in the case of improved estimation of electricity demand function by integration of fuzzy system and data mining approach. Therefore trial and error is relevant in this case. This could be highly time consuming for the web developer. By committing to the implementation of one technology, the web developer won’t know that it is successful until the end of the integration process. Choosing the wrong technology for an integration process could result in higher costs to the project and even lead to a total abandonment of doing the data integration process in the first place (Laudon, & Laudon, 2004).

In contrast, using GIWeb acted as a surveying tool removing any human bias toward a particular technology choice. The web developer was able to eliminate various options early on in their integration process narrowing down the available technology options to implement.

A limitation of the GIWeb approach in this case is tied to the web developer correcting identifying the characteristics of the web application. If the web developer does this incorrectly, this will lead to an invalid integration technology suitability match from following GIWeb. This leads to an opportunity for further research in the field to expand how web developers could better identify the characteristics of their web application.
5.7.5 The web application

This section presents the source code of the web application and its XML output. See appendix 3 for source code configuration.

5.7.6 Summary

This section presented the development of a CRM system web application using the Zend Framework and an integration scenario. It examined the application of the proposed approach GIWeb described in Chapter 4 to the integration scenario. It presented how GIWeb is included in the process and implemented by the web developer. It investigated the applicability of this as a solution for aiding the web developer in achieving their integration outcome.

The web development process was extended with the application of GIWeb. It included the web developer consulting the framework tree presented in Section 4.3.2 through a plug-in. By following this framework tree, the web developer classified the web application against the criteria checks of the framework and ensured the system and integrator requirements were best achieved and validated. The classification process examined the web application in relation to the target data source and which integration technologies best fit the integration process. The GIWeb plug-in is an extension to the Zend Framework IDE that the web developer used to develop the web application. The GIWeb plug-in presented a series of criteria checks for the web developer to contrast the characteristics of the web application against. Two integration technology options were appropriate, Remote Procedure Call (RPC) and a direct XML based integration.

Following the GIWeb revealed that a direct XML based integration was the most appropriate integration technology for the web developer to follow. This is because the RPC option replied on the external server being queried in real time. This option was ruled out due to this characteristic being unachievable in the integration scenario discussed.
The application of GIWeb has enabled the web developer to eliminate a potential integration technology as an approach to their integration process and have confidence in the decision to implement a direct XML based integration as the chosen approach. By examining the characteristics of the web application and contrasting the characteristics of suitable integration technologies, the development process has been improved saving time from potentially developing the wrong data integration approach. It has provided the ground work for mapping out and justifying the most appropriate data integration path to develop. It ensured that the merits of a potential data integration technology were assessed by its characteristics to address the system and integrator requirements by applying GIWeb.

5.8 Summary

This chapter illustrated how the proposed approach GIWeb was used to support a variety of case studies. The case studies that were chosen covered a range of development scenarios and different types of integration environments.

Section 5.2 discussed a programming environment called the Zend Framework. A scenario was described where a web developer builds a web application using the Zend Framework that requires integration with multiple data sources. APITgility was used which is a Zend Framework API. This section discussed the challenges to integrate with the target data sources. Section 5.3 presents the application of GIWeb as a solution to narrow down appropriate integration technologies. It discussed the application of GIWeb through the means of a plug-in. This plug-in enabled the web developer to classify the web application characteristics to choose between the technology options. In the scenario discussed, the web developer was able to test the appropriateness and readiness for using REST or RPC as the integration technology with an API. The GIWeb plug-in was merged with the API by editing the API configuration file. The implications included a technical learning curve in setting up the plug-in. It also involved extra thought and planning for the data integrator to implement during their integration process.

Section 5.4 examined the application of GIWeb to a project management framework PRINCE2 where the conceptual management of an integration project is modeled out. In
this scenario a project manager adopted the PRINCE2 framework as the model to follow governing all aspects of the projects implementation. There are different technology options for how data sources are to be integrated with through the PRINCE2 workflow.

Section 5.5 examined the application of GIWeb to an incident management system using the project management framework PRINCE2. In this scenario a project manager adopted the PRINCE2 framework as the model to follow governing all aspects of the projects implementation. There are different technology options for how data sources are to be integrated with through the PRINCE2 workflow. GIWeb is realised through a plug-in which acts as an extension to the PRINCE2 model. The project manager added the plug-in to the data integration component of the PRINCE2 workflow. The web developer followed the workflow defined in the PRINCE2 model using the plug-in to improve their selection of the most appropriate integration technology for the integration process. The application of the plug-in to a PRINCE2 project management software package such as CorePM involved additional preparation and planning time for a project manager.

Section 5.6 examined the application of GIWeb to Eclipse, an integrated development environment (IDE). A web developer uses Eclipse to develop a web application that is developed in PHP. This section examined the database selection process using Eclipse. The application of GIWeb created a fine tuned selection of data sources and assisted in connecting to these data sources. The process of following the GIWeb plug-in helped the data integrator generate a fine tuned list range of appropriate data sources for their integration process. It enabled the data integrator to load up a narrowed down list of appropriate data sources as an extension to the default data sources on the Eclipse framework. The GIWeb plug-in is merged with the Eclipse Framework by editing the manifest file, plugin.xml. This defined how the GIWeb plug-in extends the Eclipse platform and is parsed when the plug-in is loaded into Eclipse. The implications included the GIWeb plug-in involving extra thought and planning. It is an additional step to their development process when using Eclipse requiring additional time to prepare and implement. The GIWeb plug-in required extra coding to configure the Eclipse framework plugin.xml script.

Section 5.7 presented the development of a CRM system web application using the Zend Framework. It examined the application of GIWeb to the integration scenario. It
presented how GIWeb is included in the process and implemented by the web developer, enabling them to eliminate a potential integration technology as an approach to their integration process and have confidence in the decision to implement a direct XML based integration as the chosen approach. It provided the ground work for mapping out and justifying the most appropriate data integration path to develop. It ensured that the merits of the chosen data integration technology were assessed by its characteristics to address the system and integrator requirements from applying GIWeb.
6 Evaluation

6.1 Overview

This thesis has presented GIWeb as a solution for supporting data integration for web applications instantiated through a framework. This approach aims to improve on the existing approaches for supporting the integration of data sources for use in web applications (see Chapter 2). Current approaches are not usable as a generic solution in all integrating situations. There is a lack of generic frameworks that take into account new and emerging models and technologies. There is a lack of maturity and reusability of current approaches, compatibility issues, reliance on a technical expert and high cost to maintain. These approaches are not appropriate for all scenarios and they are broadly domain specific (see Sections 2.4, 2.5 and 2.6).

This chapter presents an evaluation of the GIWeb approach for supporting the integration of data sources for use in web applications, as proposed in Chapter 4. Section 6.2 investigates the application of GIWeb contrast against the integrator and system requirements introduced in Section 3.5 and evaluates how GIWeb supports these requirements. An evaluation of this framework is done using several case studies. This chapter evaluates why a software engineering methodology is used for this research. It examines how a web developer can develop a web application to meet the requirements discussed (see Section 3.5). It provides an evaluation examining the qualitative and quantitative outcomes of how GIWeb will meet these same requirements.

Section 6.3 examines the case scenarios presented in Chapter 5; evaluating the challenges faced when integrating using some of the approaches discussed in chapter 2 and then an evaluation of GIWeb as a solution. The evaluation of these cases examines the application of GIWeb as a solution to the difficulties associated with data integration discussed. An analysis draws conclusions on the approach and its usefulness as a generic high-level solution.
The remainder of this chapter is as follows. Section 6.2 argues that the application of GIWeb in the cases presented in Chapter 5 meets the requirements described in Section 3.5 evaluating the qualitative outcomes. Section 6.3 provides an analysis of the quantitative outcomes. Section 6.4 discusses GIWeb and the goals it achieved. Section 6.5 summarises conclusions on the above sections.

6.2 Evaluation methodology

An evaluation is needed to determine the applicability of GIWeb as a solution for supporting data integration for web applications instantiated through a framework. It is designed to build upon the results of the cases discussed in chapter 5 and undertaken to clarify and extend the results of these cases. The results of the evaluation will be used to determine the merits of applying GIWeb by web application builders in their development process when integrating data sources.

The evaluation process is based around comparing if the proposed approach, GIWeb, supports the system and integrator requirements proposed in Chapter 3. Extending on the requirements groups proposed by (Sommerville, 2001) they will determine what a framework for data integration should do to as well as formalise what requirements can be followed in a data integration process to identify what is needed to successfully achieve a data integration goal. The results will be based on assessing qualitative outputs and quantitative outputs.

Acceptance of an integration technology, decisions based on following GIWeb and the level of quality in supporting a data integrator will be examined. Quantitative characteristics examined include examining number of clicks, quality outputs, number of data sources available, number of data integration technologies to consider, technical outcomes eliminated from following the approach and the number of checks before making a decision. The guidelines to follow to implement the evaluation include ensuring the approach meets the system and integrator requirements identified in Chapter 3 and to examine the quantitative performance and quality aspects of the integration selection options.
6.3 Qualitative evaluation

6.3.1 Overview

This section presents a qualitative evaluation of GIWeb, the approach proposed in Chapter 4, for supporting the integration of data sources for use in web applications. It examines the requirements presented in Chapter 3 and how GIWeb meets these requirements. It investigates the suitability of GIWeb in supporting varied application. It contrasts it against the integrator and system requirements described in Section 3.5 to evaluate the validity of GIWeb supporting the development of applications which integrate different data sources.

6.3.1 How integrator requirements have been met

The data integrator requirements are the requirements that a user who is attempting the integration process would expect to see. There are two integrator requirements described in Section 3.5.3.

Requirement 1: The process of integrating data sources for a web application must provide a means of accessing the data from the various sources in a consistent, timely and accurate manner.

Requirement 2: The ability to implement, manage and control the integration process without the ongoing reliance on a third party or highly skilled technical person.

As discussed in Section 3.5.4 these requirements are from the perspective of a data integrator such as a web developer. It is what they would expect as an ideal outcome from the process. Chapter 4 proposed GIWeb, a methodology instantiated through a framework. The framework acts as a support mechanism for a data integrator, enabling them to choose the appropriate integration approach to implement.

The support framework presents a series of questions based on determining the characteristics of a web application and expected integration outcome as discussed in
Section 4.3.2. These questions allow a data integrator to contrast their web application against the framework criteria and classify it under a specific category of web application. As described in Section 4.3, the support framework is split into three phases:

Phase 1: Identify characteristics of the web application

Phase 2: Compare these to characteristics of suitable approaches

Phase 3: Identify appropriate technology to implement for integration process

These phases enable a data integrator to choose an appropriate integration technology to use in their integration process by comparing the different technology choices against the requirements characteristics described in Chapter 3. Some of these characteristics from the framework specific to the integrator requirements include assessing sources be queried in real time, contrasting if the technology approach is centralised or decentralized and if is self-manageable.

The data integrator can further narrow down appropriate technology choices using the framework criteria checks. They can analyse the various dimensions of an integration technology to assess if one might be the most appropriate technology choice for their integration process. They can use the framework criteria checks to evaluate the different technology choices against the outcomes they would expect to see from the integration process.

By following the framework structure, the data integrator can ensure that the technologies chosen achieve live data access with their target sources, achieve this in a fast and accurate manner and the integration is consistent and accurate between target sources. The data integrator can ensure that the chosen technology enables them to implement, manage and control the integration process without the ongoing reliance on a third party or highly skilled technical person.

Chapter 5 highlighted a case where a Customer Relationship Management web application was developed using the Zend framework and configured using GIWeb. In this case, the data integrator surveyed the different technologies based on their
characteristics to determine the technical outcomes of each type of technology. These options included using an API, message queueing, program-to-program communication, direct XML and JSON. Each one has a different expected technical outcome and by using the framework, the data integrator could see which outcomes align with their expectations for the integration process. In this case, the options were narrowed down from five to two: Remote Procedure Call (RPC) and a direct XML based integration incorporating the same data source. Only RPC and direct XML were sufficient for connectivity, architecture and ease of use in this case given the specific formatting required and the required architecture and limitations with the external server. The application of GIWeb enabled the data integrator to compare the different characteristics of each option for accessing the data from the various sources in a consistent, timely and accurate manner.

The data integrator expects to control the integration from various sources independently of a third party or highly skilled technical person. Three of the approaches identified by the framework require direct database access. This means they might not fit the expected outcome for timely integration from the various data sources. There will need to be considerations around establishing direct access to the data sources and if there are connection issues or other technical concerns around the accuracy of data via different connections. The alternative is to make use of an API for accessing data sources for real time data access and data accuracy or a pre-made API specific for data integration purpose. Four of the approaches are not self-manageable. While the last two options query the data in real time, the support framework enables the data integrator to further narrow down the options contrasting a deeper level of criteria (see Figure 4.10). In this scenario it is determined that one technical approach is more widely used and has higher support than the other approach. The data integrator can therefore determine this final approach is the one best aligned with their expected outcomes. A UML Activity Diagram illustrates this workflow in Figure 6.1.
In Figure 6.1, the combined phases of the framework enable the data integrator to achieve both requirements. They can ensure the technologies evaluated achieve the desired process of integrating data sources for the purposes of presenting the data in a
web application with a means of accessing the data from the various sources in a consistent, timely and accurate manner. They can also ensure the chosen technologies are appropriate to implement, manage and control the integration process without the ongoing reliance on a third party or highly skilled technical person.

Section 5.3 presented a case where an incident management web application was developed using the Zend framework. The web developer was able to test the appropriateness and readiness for using REST or RPC as the integration technology. GIWeb enabled the web developer to classify the web application characteristics to choose between the technology options. In this case, this involved deciding if the web application is programmable or not, if it will involve real time querying, translation, if it involves using meta-data and if it is self-manageable. In this case, an outcome was that REST was shown to be more appropriate because it was narrowed down to be more self-manageable from the two options.

Section 5.5 examined the application of GIWeb to the development of the incident management system using the project management framework PRINCE2. GIWeb is realised through a plug-in which acted as an extension to the PRINCE2 model. The project manager added the plug-in to the data integration component of the PRINCE2 workflow. By applying GIWeb in this case, it resulted in a new quality MP Output. As discussed in Section 5.5, the addition of the MP Output called Integration Framework effectively measures and ensures the quality of the integrity of the chosen integration technology because the process has ensured a proper analysis of options and measuring them against the GIWeb criteria checks. Regardless of which technologies the web developer implements during the data integration stage of the project, the project manager has enhanced the process through a validation and verification workflow in choosing which technology to implement. The quality process was enhanced from the application of GIWeb by narrowing down technology options based on assessing their characteristics for being consistent, timely and accurate as well as their self-management suitability.

Section 5.6 presented a case using Eclipse to develop a quoting web application that is developed in PHP and will be part of a mobile app. Implementing GIWeb in this case assisted the data integrator by in generating connection specific code for different data
sources. Once the GIWeb plug-in was loaded, the data integrator was able to select from the relevant connection profile as part of the data source connection wizard in Eclipse. The implementation of GIWeb created a fine tuned selection of data sources and assisted in connecting to these data sources. The process of following the GIWeb plug-in helped the data integrator generate a narrowed list range of appropriate data sources for their integration process based on assessing their characteristics in line with the integrator requirements for being consistent, timely and accurate as well as their self-management suitability.

Without applying GIWeb in these cases, the alternative is that the data integrator would be forced to a trial and error in applying different integration technologies to each of the integration processes discussed. This could be very time consuming for the data integrator. By committing to the implementation of one technology, they won’t know that it is successful until the end of the integration process. They could choose a technology that results in inconsistent, untimely and inaccurate result or relies on management and control of the integration process by a third party or highly skilled technical person. Choosing the wrong technology for the integration process would result in higher costs to the project and even lead to a total abandonment of doing the data integration process in the first place. Section 6.4 Quantitative Outcomes will examine in greater detail these alternatives.

6.3.2 How system requirements have been met

The system requirements are technical outcomes that need to be supported in an integration process. As described in Section 3.5.3.2 there are four aspects, Design Process, Heterogeneity, Structure and Integrity Support and Performance with each of these focusing on the technical considerations in each area. Section 6.2.3 evaluates how GIWeb helped support these types of requirements through the cases presented in Chapter 5 and examines the outcomes for meeting these requirements without GIWeb.
6.3.3 Design process system requirement

The analysis, classification and development phases are an aspect of GIWeb that supports the design process requirement. The architecture being integrated with is identified at this stage. The data integrator is encouraged to set a goal of modeling the data integration landscape that is to be implemented. By doing so, they create an overall picture of what the structure will look like, what the expectations are and an architecture to base their integration process on. It enables a data integrator to map out the right path to follow and it provides a structured design process.

In the case discussed in Section 5.7.4, the data integrator followed GIWeb beginning by determining the goals, motivation, purpose and definition of their integration activity. The data integrator does this by collecting user requirements and the data needed to be presented to the user. In the cases discussed in Section 5.3, Section 5.5 and Section 5.6, the system requirements were met with GIWeb through the analysis and classification phases. They identified who the user is, assign priorities to each requirement, determining data sources and determining web applications being integrated. By following these activities, the data integrator has created a conceptual design of the overall data integration architecture between the different web application data sources and web applications being integrated. This aided the data integrator in the case described in mapping out different paths to follow and providing a structured design process for the data integration process.

If GIWeb wasn’t applied in these cases, the web development process described in each case didn’t involve describing the architecture being integrated with, modeling the integration landscape, validating the technical requirements for the implementation of the integration process. By not doing these activities, the system requirements are not met in these cases. Applying GIWeb aided the data integrator in achieving these requirements.

To illustrate this further, in the scenario in Figure 6.2 it shows a UML Component Diagram of a data integrator following the analysis, classification and development phase of the methodology by mapping out the integration architecture describing the
transformation between the data sources and database targets. The data sources range from a customer relationship management system, financial system and enterprise reporting system. The database targets range from a MySQL database and a Cache database. An extraction stage is included to describe the extraction of the data from the data sources into the database targets. This consists of sorting records, cleansing data and validation. The later stage of the transformation consists of joining data, sorting dimensions, merging fields and aggregation.

![UML component diagram mapping out integration process architecture](image)

Figure 6.2 UML component diagram mapping out integration process architecture

### 6.3.4 Heterogeneous system requirement

In the cases discussed in Chapter 5, GIWeb enabled a data integrator to choose the most appropriate technical approach for their integration process. Using the support framework described in Section 4.3.2 the data integrator in each of the cases was able to determine the characteristics of a particular approach to determine its heterogeneity.
In Section 5.6, the implementation of GIWeb assisted the web developer by in generating connection specific code for different data sources. In the case discussed in Section 5.7 support for heterogeneous environments was enhanced through the selection process. The data integrator was able to contrast the technical outcomes of different approaches using the support framework to examine support for heterogeneous environments, integration from a variety of sources consisting of different structures and formats. This could include different file formats, access protocols and query languages, different ways of representing and storing the same data and even relationships between data. Figure 6.3 illustrates this workflow with a UML Activity Diagram.

![Figure 6.3 UML activity diagram of data integrator evaluation of technology characteristics for heterogeneity](image)

In the alternative to applying GIWeb to these cases, the support for heterogeneous environments is left to a level of guess work by the data integrator. With no verification or validation process in the web development process described in each of the cases, the decision making process lacks the vetting assessment before deciding on a technology to choose. This could result in erroneous choices for a data integration technology
integration when dealing with a variety of sources consisting of different structures and formats. Section 6.4 Quantitative Outcomes will expand on these outcomes.

6.3.5 Quality system requirement

Structure, integrity support and performance are a requirement met through the quality phase. In this phase the integration process support data quality and data governance. System technical outcomes examine the quality of the integration process performance. In the cases discussed in Chapter 5, this phase enabled the data integrator to examine the speed of the data transformation and presentation between sources. Measuring the ability to handle unpredictable data and testing the ongoing integration of data sources is successful. The quality of the performance can be both its current performance for an integration activity and its ongoing performance into the future for different integration activities such as evolving data sources.

The quality phase examined the accessibility and uptime of the integration activity and the reliability to integrate with data sources in each of the cases in Chapter 5. If the data sources can’t be integrated with due to downtime, the accessibility measure fails. In Section 5.6 the data integrator was able to assess each of the potential integration technologies against these criteria when using GIWeb for the appropriateness and readiness for using REST or RPC. Testing the continued accessibility of the integration with data sources is a core quality indicator of a chosen technology approach and this was a characteristic that contributed to narrowing down the technology options in each of the cases. As seen in Section 5.7.4 where implementing the GIWeb Plug-in as an extension to Zend Framework, data translation and accessibility were contributing characteristics to check for in verifying suitable technology options. The data format may change from one source to the other as part of the integration process but the information contained should be presented the same without any loss or misinterpretation from what as in the original source. Figure 6.4 illustrates this workflow with a UML Activity Diagram. The system technical outcomes of these areas combined enable the data integrator to validate the quality of a chosen integration technology for an integration process. In the case discussed in Section 5.5, adding GIWeb as a plug-in to the data integration component of the PRINCE2 workflow resulted in a new quality
process MP Output at the cost of an additional work package as shown in Figure 6.5. A process in PRINCE2 is managing product delivery (MP) and a quality MP output is determined which is labelled approach options as shown in Figure 5.12. The project manager was able to satisfy quality checks in the PRINCE2 workflow knowing that the data integration component had gone through a verification and validation process.

Figure 6.4 UML activity diagram of data integrator evaluation of quality outcomes
Figure 6.5 Bar graph comparison of PRINCE2 model number of MP outputs and work packages

The alternative to applying GIWeb in these cases is that the quality system requirement isn't met with the web development process described in each case. In the case discussed in Section 5.5, with the model lacking the GIWeb Plug-in it would mean the project manager has a reduced quality output in the MP Outputs and therefore an increased risk in the project. The project manager now faces the risk of an incorrect integration technology being chosen by the developer because they have not gone through an evaluation of the technical outcomes examining the data quality potential of each integration option. In the case discussed in Section 5.7.4, without GIWeb there is an increased potential for human bias when choosing a particular technology choice. The web developer doesn’t eliminate various options early on in their integration process narrowing down the available technology options to implement. This leaves only a trial and error scenario in applying different integration technologies to the integration process. By committing to the implementation of one technology, the web developer won’t know that it is successful until the end of the integration process. This could lead to data quality and data governance issues as well as loss or misinterpretation from what was in the original data source. When GIWeb was applied to each of the cases, the data integrator was aided in their technology selection process by eliminating potential technologies that didn’t survey well based on their data quality characteristics.
6.3.6 Performance system requirement

The test phase and the quality phase enable the data integrator to ensure performance of the data integration process. A performance indicator is that there is no lag in presentation or loss in quality of data. This ultimately is experienced by the data integrator once they've implemented their integration process so it is vitally important that the chosen integration technology be the most appropriate to avoid lag or loss in data quality. The technology chosen for an integration process should have the ability to recover from errors such as source or target systems going down, data quality issues.

When GIWeb was applied in the cases discussed in Chapter 5, the test phase validated the technical integration method and the flow of the data between data sources. It involved testing the components of the integration activity to ensure they work as intended and achieve the expected technical outcomes such as the web application script, the target data sources, the data that is presented to the user from the data sources and the validity of this data.

Using GIWeb, the data integrator measured the technical outcomes that need to be supported in an integration process. In the case discussed in Section 5.7, the system requirements of design process, heterogeneity, structure and integrity support and performance are achieved by implementing GIWeb. Section 5.7.4 showed that applying GIWeb consisted of examining the available technology options for maintaining optimal performance when integrating between a data source to its destination. In this case, when the data integrator progressed through the GIWeb framework tree, the technology options of using an API, message queueing, program-to-program communication, direct XML and JSON were each cross checked against performance characteristics such as connectivity, ease of use, readiness and expandability. The case presented in Section 5.6 showed the application of GIWeb as a plug-in to Eclipse enabling the web developer to compare the criteria checks presented in Section 4.2.2 and Section 4.3.2 to narrow down the suitable data sources for their web application. This included validating that each data source is rated better based on its characteristics for maintaining optimal
performance thereby eliminating inappropriate sources that rated highly in lag in presentation or loss in quality of data.

The alternative to applying GIWeb in these cases is that the performance requirement is not met in the original case scenario when applying the web development process described in each case. GIWeb enabled a comparison of the performance characteristics of each potential integration technology to narrow down suitable options. If this step was not implemented in the cases, the performance outcomes would not have been known to the data integrator until a chosen integration technology was chosen and implemented. By this stage, an erroneous choice could be fatal to the overall outcome of the project or result in increased costs and time trying to re-do the integration process with an alternative integration technology option. Section 6.4 Quantitative Outcomes will examine in greater detail these erroneous choices. Another outcome could be an underperforming integration technology which could have roll on performance effects to the project such as data loss, high difficulty of use or limited expandability.

6.4 Quantitative outcomes

6.4.1 Overview

This section evaluates the quantitative outcomes of GIWeb proposed in Chapter 4 for supporting the integration of data sources for use in web applications. It examines the statistics of current approaches described in Chapter 2 compared to GIWeb. It compares the benefits and disadvantages of the approach proposed and analyses how well it achieves its goals. This section also discusses the implementation of GIWeb and its implications.

6.4.2 Quantitative analysis

This section examines the statistics of existing approaches as a solution to the goals discussed in this thesis in comparison to GIWeb. It looks at areas such as usage, consistency, difficulty and speed and measures the appropriateness of each of these
This section discusses GIWeb and its application to the cases in Chapter 5 compares the outcomes of not applying GIWeb as a solution. The other approaches that a data integrator could implement range from selecting a model to follow, a technology to implement or a management process to guide their integration process (see Section 2.4, Section 2.5 and Section 2.6). GIWeb helps the data integrator determine which model, technology or management process-based approaches is the most appropriate one to implement for their specific integration process. Contrast to a data integrator simply selecting a model, technology or management process-based approaches on its own and not knowing if it is suited to their integration process. Choosing an appropriate data integration approach and technology is an important step in the web development process which requires deep knowledge of many possible approaches and technologies. The consequences of choosing the wrong approach or technology could result in the data integration failing or only achieving a partial success, poor quality data results or incorrectly matched data fields (see Section 2.5). GIWeb enables the data integrator to narrow down the suitable options and validate their decision on what model, technology or management process-based approaches is appropriate for their integration process by following the framework. GIWeb may also help the data integrator determine that extending an existing approach, such as extending a model, is an appropriate solution for their integration process.

GIWeb’s framework was discussed in Chapter 5 in several case scenarios. It was realised as a plug-in. The following section discusses a quantitate evaluation of the proposed approach to illustrate the effect of the solution.

### 6.4.3 Simplification of choices analysis

This section focuses on an evaluation of the performance of GIWeb. Chapter 5 discussed the application of GIWeb to a wide range of scenarios examining different types of integration environments. These included programming environments such as development using the Eclipse platform and Zend framework and the application of the
approach to the project management framework PRINCE2 where the conceptual management of an integration project is modeled out.

### 6.4.3.1 Integration selection options

To evaluate the effectiveness of the GIWeb plug-in, the integration selection options in the process the data integrator goes through both before and after the use of the plugin are contrasted for each of the cases discussed in Chapter 5.

Section 5.3 presented a case where an incident management web application was developed using the Zend framework. In this scenario, the data integrator has to make a decision to determine if Rest Services or RPC Services were the right integration path to take for their integration process. The application of GIWeb’s framework to this scenario involved implementing a plug-in as an extension to the Zend Frameworks APIgility solution. The GIWeb plug-in involved using a wizard to categorise the web applications characteristics. To evaluate the effectiveness of the plug-in, the characteristics of the process the data integrator goes through both before and after the use of the plugin are contrasted. The number of clicks examined, assumes the minimum number of required clicks to complete the task. The positive and negative effects of their performance are then reviewed.

The process a data integrator went through when using APIgility as discussed in Section 5.3 involved the data integrator choosing between REST Services or RPC Services. Table 6.1 shows the current number of clicks the data integrator went through when determining the API type used for their integration process. It also shows the percentage chance of the data integrator picking the wrong API type during this selection process when using APIgility.

<table>
<thead>
<tr>
<th>Process</th>
<th>Number of clicks</th>
<th>Chance of selecting the wrong option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current APIgility Wizard</td>
<td>1</td>
<td>50%</td>
</tr>
</tbody>
</table>

Table 6.1: APIgility number of clicks versus chance of selecting wrong option
The GIWeb plug-in changed this process so that the data integrator first goes through the wizard and its process to filter which of the two service options is the most appropriate to choose.

Table 6.2 shows the current number of clicks the data integrator goes through when determining the API type used for their integration process when using the GIWeb plug-in extension to APIgility. It shows the percentage chance of the data integrator picking the wrong API type during this selection process when using the GIWeb plug-in extension to APIgility.
<table>
<thead>
<tr>
<th>Process</th>
<th>Number of clicks</th>
<th>Chance of selecting the wrong option</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIWeb Plug-in Extension to APIgility Wizard</td>
<td>3</td>
<td>1%</td>
</tr>
<tr>
<td>Extending existing frameworks</td>
<td>&gt; 3</td>
<td>50%</td>
</tr>
<tr>
<td>Adapting an IT management framework</td>
<td>&gt; 3</td>
<td>50%</td>
</tr>
<tr>
<td>Altering toolkits</td>
<td>&gt; 3</td>
<td>50%</td>
</tr>
<tr>
<td>Not applying GIWeb plug-in extension to APIgility</td>
<td>1</td>
<td>50%</td>
</tr>
</tbody>
</table>

Table 6.2: Extension to APIgility comparison

This use of the GIWeb plug-in is an extension of the management process-based approaches altering software development frameworks with the data integrator altering the APIgility software framework to implement the plug-in. There is a decrease in the number of clicks to choose an API type when using the current APIgility process. However, it is argued there is a 50% chance that a data integrator can pick the wrong API type between the two options REST Services or RPC Services.

A first time user who is unfamiliar with which API type is most appropriate has to choose which of these two options is suitable for their integration process leaving an element of risk in choosing the wrong option. In the instance they do have prior knowledge of which type to use, Chapter 2 highlighted how these can be unknown at design time, it may still not be the most appropriate type for their integration process and ultimately be the wrong selection in such scenario as well. In contrast to other approaches such as extending existing frameworks, adapting an IT management framework or altering a toolkit, each of these require a greater number of clicks for the data integrator to follow to implement. By not applying the GIWeb plug-in extension, the number of clicks is the lowest at 1 click but the chance of selecting the wrong option is at 50%.
Current approaches discussed in Chapter 2 such as IT management frameworks do not respond to unique environments. They are too general and too complex and have overlapping (see Section 2.6). It is the combination of these areas that result in an increased number of clicks to implement and a higher percentage chance in selecting the wrong option as illustrated in Table 6.3.2. Extending frameworks are often designed to be a one size fits all approach (see Section 2.6) with programs developed using multiple frameworks are prone to have interaction problems between them resulting in the same outcome as shown in Table 6.2. With altering a toolkit, the data integrator has an additional decision making process to determine which toolkit to use for their integration process. It is also not a centralised approach or self-manageable because of its complexity. The complexity of implementing a toolkit results in a higher level of clicks and a higher percentage chance in selecting the wrong option.

With the application of the GIWeb plug-in to this scenario, the number of clicks increases with the data integrator force to go through a web application categorisation process as discussed in Section 5.3. The outcome of this is a reduced chance in selecting the wrong API option for their integration process. It is argued that the chance of selecting the wrong option has decreased significantly to as low as 1% with still the chance the data integrator decides not to follow the recommended option or has incorrectly categorised their web application during the process.

The outcome of reducing the chance of selecting the wrong API option for the integration process outweighs the increased number of clicks to ensure this. The GIWeb plug-in aids the data integrator as a value added approach to ensuring the selection of the right API path for the integration process. This scenario was an example of narrowing down the approach options for the data integrator.

Section 5.5 examined the application of GIWeb to the development of the incident management system using the project management framework PRINCE2 and the project management software CorePM. In this case the project manager added the GIWeb plug-in to the data integration component of the PRINCE2 workflow. The web developer who follows the workflow in the PRINCE2 model will use the GIWeb plug-in to improve their selection of the most appropriate integration technology for the integration process. In the case, the CorePM product was used and the workflow was extended by
including the activities of the GIWeb plug-on as a new process in the implementation stage. It lead to justification in the decisions made when implementing the work package for the Integration process. By following the proposed frameworks workflow, a validity check has been performed to justify the integration process chosen. The result enabling them to issue the appropriate integration process work packages and when completed, an integration plan is one of the MP outputs.

This is an outcome from being narrowed down through the GIWeb criteria checks. This output effectively measures and ensures the quality of the integrity of the chosen data integration technology because the process has ensured a proper analysis of options and measuring them against the GIWeb criteria checks.

Table 6.3 shows the current number of MP Outputs, Controlling Stage Workflows and change in MP quality when considering the application of GIWeb to the PRINCE2 model when using CorePM as discussed in Section 5.5.

<table>
<thead>
<tr>
<th>Process</th>
<th>Number of MP Outputs</th>
<th>Controlling Stage Work Flows</th>
<th>Increased Quality at MP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation of GIWeb plug-in</td>
<td>4</td>
<td>3</td>
<td>100%</td>
</tr>
<tr>
<td>Not implementing the GIWeb plug-in</td>
<td>3</td>
<td>2</td>
<td>No change</td>
</tr>
</tbody>
</table>

Table 6.3: PRINCE2 CorePM outputs from the application of GIWeb

Increasing the number of MP Outputs in the example using CorePM reduces the project outcome risk.

The alternative to applying GIWeb in this case is that the project manager has a reduced quality output in the MP Outputs and therefore an increased risk in the project. There is risk of an incorrect integration technology being chosen by the developer because they have not gone through an evaluation of the technical outcomes examining the data quality potential of each integration option. The web developer doesn’t eliminate various options early on in their integration process narrowing down the available technology options to implement. This leaves only a trial and error scenario in applying different
integration technologies to the integration process. By committing to the implementation of one technology, the web developer won’t know that it is successful until the end of the integration process (see Chapter 2).

Section 5.6 presented a case using Eclipse to develop a quoting web application that is developed in PHP and will be part of a mobile app. Several data sources are being integrated with. The target data sources in this example are outside the range of the default list of data sources in Eclipse’s database selector. The application of GIWeb to this scenario involved implementing the GIWeb plug-in as an extension to the Eclipse Framework. The GIWeb plug-in involved using a wizard to categorise the web applications characteristics.

The aim of the wizard is to enable the web developer to compare the criteria checks presented in Section 4.2.2 and Section 4.3.2 to narrow down the suitable data sources for their web application. The process of following the wizard helps the data integrator generate a fine tuned list range of appropriate data sources for their integration process but also extends the available data source options and at the same time. The process ensures they have chosen the appropriate data source option for their integration process and extends the available options to integrate with. The current process a data integrator goes through when using the Eclipse Framework involves the data integrator choosing from a static list of data sources.

Table 6.4 shows the current number of data sources the data integrator has available to them when using the Eclipse Framework.

<table>
<thead>
<tr>
<th>Process</th>
<th>Number of data sources available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current Eclipse Data Source Wizard</td>
<td>10</td>
</tr>
<tr>
<td>Implementation of GIWeb plug-in</td>
<td>4</td>
</tr>
<tr>
<td>Not implementing the GIWeb plug-in</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 6.4: Eclipse framework number of current data sources

The number of available data sources increases once the GIWeb plug-in has been implemented with the Eclipse Framework. Table 6.4 also shows the decreased number
of data sources available once the GIWeb plug-in is implemented. The original data sources remain with the extension enabling the addition of any new specified ones from the GIWeb plug-in process. The web application characterisation process that the data integrator goes through with the GIWeb plug-in enables them to narrow down appropriate data sources. From this narrowed down list, additional ones outside the default list currently in Eclipse may be identified. When configuring the plugin.xml file, the data integrator can expand on this list as needed. The outcomes examined highlighted a positive outcome from using the GIWeb plug-in resulting in the expansion of available data sources for the data integrator during their integration process.

Without applying GIWeb in this case, the alternative is that the data integrator would be forced to a trial and error in applying different integration technologies to each of the integration processes discussed. Figure 6.6 highlights this comparison showing the number of attempts versus potential errors when comparing the application of GIWeb to the case versus not applying it.

![Number of Attempts vs Potential Errors](image)

Figure 6.6 Bar and line graph comparison of number of attempts versus potential errors

The increased number of attempts and potential errors could be very time consuming for the data integrator. By committing to the implementation of one technology, they won’t
know that it is successful until the end of the integration process. They could choose a technology that results in inconsistent, untimely and inaccurate result or relies on management and control of the integration process by a third party or highly skilled technical person.

Section 5.7 presented a case where a Customer Relationship Management web application was developed using the Zend framework and configured using GIWeb. It involved implementing the GIWeb plug-in as an extension to the Zend Framework IDE that the web developer used to develop the web application. The GIWeb plug-in presented a series of criteria checks for the web developer to contrast the characteristics of the web application against. By following this process, it narrowed down the available integration technology options as illustrated in Table 6.5.

<table>
<thead>
<tr>
<th>Process</th>
<th>Number of Data Integration Technologies to consider</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation of GIWeb plug-in</td>
<td>2</td>
</tr>
<tr>
<td>GIWeb is not applied to the case</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 6.5: Integration technology approach options comparison

After following the criteria checks from the framework, it was evident that two technology approaches were appropriate, being Remote Procedure Call (RPC) and a direct XML based integration.
Following the framework revealed that a direct XML based integration was the most appropriate integration technology for the web developer to follow. This is because the RPC option relied on the external server being queried in real time which was not a suitable technical outcome in this specific integration process. Following the framework meant the data integrator was able to examine and eliminate this technical approach as an appropriate choice because of this technical outcome.

<table>
<thead>
<tr>
<th>Process</th>
<th>Technical Outcomes Eliminated from following GIWeb</th>
<th>Remaining Technical Options to Consider</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation of GIWeb plug-in</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Not implementing the GIWeb Plug-in</td>
<td>0</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 6.6: Narrowed down technology approach options from GIWeb plug-in

Following the framework via the GIWeb plug-in enabled the web developer to eliminate a potential integration technology as an approach to their integration process and have confidence in the decision to implement a direct XML based integration as the chosen approach. In contrast by not implementing the GIWeb plug-in, no outcomes could be eliminated from the framework.

6.4.3.2 Decision making

Chapter 2 highlighted the implications of choosing the right integration approach in an integration process. The outcomes of choosing an in-appropriate integration technology include lack of support, lack of compatibility or even total failure in the data integration process. GIWeb has a value added benefit of aiding the data integrator in helping narrow down which integration option to select based on their specific integration process and having the confidence in that decision based on a thorough checking process.
Section 5.5 examined the application of GIWeb to the development of the incident management system using the project management framework PRINCE2. Its aim being to fine tune the integration component in the PRINCE2 workflow by enabling the web developer to follow the workflow defined in the PRINCE2 model will use a GIWeb plug-in to improve their selection of the most appropriate integration technologies for their integration process.

As discussed in Chapter 5, the GIWeb plug-in becomes a task to follow when the part of the workflow it belongs to is executed. The workflow is extended by including the activities of the plug-on as a new process in the implementation stage. This is where the data integrator follows the methodology and framework tree presented in Section 4.2.2 and Section 4.3.2 for their integration process. The follow on stage is to then determine the outcome of the initial criteria checks. This then leads to a narrowed down list of approach options and a decision on which approach is the most appropriate one to use.

This output effectively measures and ensures the quality of the integrity of the chosen approach option because the process has ensured a proper analysis of options and measuring them against the frameworks criteria checks. This results in a validation and verification workflow in choosing which technology to implement in the chosen approach for the project manager which subsequently is an improved project quality output.

Table 6.7 shows the current number of checks performed when using the PRINCE2 framework versus with the extension proposed. The current workflow has the integration process as a work package component of the PRINCE2 framework. It is just one of many a work package elements with no further workflow outside of the current PRINCE2 model. The project manager defines the integration process as a work package and its outcome is a MP output. There are no further decision making checks or workflow for this work package on the model.

<table>
<thead>
<tr>
<th>Process</th>
<th>Number of checks before integration process decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issue work package</td>
<td>0</td>
</tr>
<tr>
<td>Extension of PRINCE2 with GIWeb</td>
<td>&gt;1</td>
</tr>
</tbody>
</table>
The extension of the PRINCE2 framework with the GIWeb plug-in discussed in Chapter 5 would see the Project Manager in the controlling stage interacting with GIWeb to determine the appropriate integration process to follow. This is a process prior to issuing work packages enabling detailed planning and justification as to the decisions made from following the framework. By applying GIWeb, a validity check has been performed to justify the decision for the integration process chosen. The result enabling them to issue the appropriate integration process work packages and when completed, an integration plan is one of the MP outputs.

The number of steps in the PRINCE2 workflow increases with the addition of the GIWeb plug-in to this workflow. The steps that interact with the step: Control a Stage (CS) increases as shown in Table 6.8.
<table>
<thead>
<tr>
<th>Process</th>
<th>Number of steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control a Stage (CS) with the current PRINCE2 Model</td>
<td>3</td>
</tr>
<tr>
<td>Control a Stage (CS) with the extension to PRINCE2 Model</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 6.8 Number of steps at the CS stage in the PRINCE2 model

The increased number of steps in this process requires both additional planning from the PRINCE2 project manager and time to implement and manage. For example, the implementation discussed in Section 5.5 involved the project manager creating six new processes and an additional quality MP Output. These processes extend the PRINCE2 workflow with the activities described in the methodology and framework tree presented in Section 4.2.2 and Section 4.3.2. This extra planning and implementation for the project manager requires more time commitment. But the value added benefits of going through this process enables the project manager to validate and justify the decision for a particular integration process as part of their overall PRINCE2 project model.

The application of the GIWeb plug-in to the Eclipse platform discussed in Section 5.6 features a wizard that assists a web developer in choosing the particular data source they are going to implement with for their integration process. It enables a web developer to use the wizard to improve their selection of the most appropriate integration data sources for the integration process by narrowing down the most appropriate data source based on their web application characteristics. This process involves the web developer altering the plugin.xml host file. The wizard prepares the selection process for the web developer by classifying the type of integration process based on the web application characteristics. The output is a reduction in decisions from the web developer with the wizard helping to narrow down the most appropriate option to implement.

Table 6.9 shows the current number of decisions a data integrator has to make when using the Eclipse platform for their integration process. It also shows the number of decisions a data integrator has to make when using GIWeb plug-in extension for Eclipse platform for their integration process.
Table 6.9 Current Eclipse number of decisions versus extension to Eclipse

<table>
<thead>
<tr>
<th>Process</th>
<th>Number of decisions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current Eclipse Data Source Wizard</td>
<td>&gt;2</td>
</tr>
<tr>
<td>GIWeb Plug-in extension to Eclipse Data Source Wizard</td>
<td>1</td>
</tr>
</tbody>
</table>

The number of decisions required by the data integrator decreases with the addition of the GIWeb plug-in. Similar to the PRINCE2 model, additional planning and time commitment is required to implement the GIWeb plug-in but the tradeoff is a reduction in the number of decisions required during the integration process.

6.4.4 Summary

This section focused on an evaluation of the performance of GIWeb based on the evaluation on the range of cases discussed in Chapter 5 examining the different types of integration environments. The value added benefits of GIWeb included an outcome where the data integrator has a narrower selection of integration options to choose from. It helped narrow down which integration option to select based on their specific integration process and having the confidence in that decision based on a thorough checking process.

The theme across the different examples examined was an increase in preparation, planning and steps to implement the GIWeb plug-in by the data integrator for their integration process. This resulted in extra workload for the data integrator in their implementation. The positive outcome was the reduced number of decisions in choosing the appropriate integration process and adequate validation to confidentially choose a specific technology approach. By going through the framework process examining the different characteristics and criteria checks of the web application, the data integrator is able to narrow down appropriate integration technologies and validate which one should be chosen for their process.
There was a reduction in the chance of selecting the wrong integration technology. By following GIWeb, the data integrator in each case discussed in Chapter 5 was able to eliminate integration technologies that were not appropriate to their integration process. The number of available data sources was also decreased. The web application characterisation process that the data integrator goes through with the GIWeb plug-in enabled them to narrow down appropriate data sources. From this narrowed down list, additional ones outside the default list could be identified. When configuring the web application, the data integrator can expand on this list as needed to include additional data sources outside of the original default list as highlighted with the example of Eclipse.

### 6.5 Goals achieved

GIWeb improves the integration of data sources for use in web applications with an approach to solving the problems discussed in chapter 2. It contributes to improving the development of a web application by aiding the web application builder in their data integration process. It has a range of goals discussed in Chapter 4 which are summarised in table 6.10.

<table>
<thead>
<tr>
<th>Goal 1</th>
<th>A high-level methodology to support the data integration process. To be used to consult with and aid in attempting to integrate with data sources for web applications.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal 2</td>
<td>Bridge the gap between technologies and processes as an integrated technology and process approach</td>
</tr>
<tr>
<td>Goal 3</td>
<td>A methodology to support the classification scheme presented in Chapter 2</td>
</tr>
<tr>
<td>Goal 4</td>
<td>Support the requirements from Chapter 3</td>
</tr>
<tr>
<td>Goal 5</td>
<td>Acts as a high-level methodology that can be applied to all data integration scenarios.</td>
</tr>
<tr>
<td>Goal 6</td>
<td>Support maturity and reusability of current approaches</td>
</tr>
<tr>
<td>Goal 7</td>
<td>Enable the data integrator to resolve compatibility issues in an integration process through the process of choosing an implementation technology</td>
</tr>
</tbody>
</table>
Goal 8

| Enable the data integrator to choose a technology for their integration process that does not have sole reliance on a technical expert to implement and maintain |

Table 6.10 GIWeb goals

Goal 1 is achieved with GIWeb. The GIWeb methodology is a high-level goal centric approach with a top-down design where data integration involves working out the goal first, identifying where the web application fits, applying a methodology and importing the technology to apply it. It supports the data integration process by being used to consult with and aid in integrating with data sources for web applications.

Goal 2 is achieved through GIWeb combining a methodology with a framework. This approach is a combined technology and process approach linking the modeling of an integration landscape to the technologies implemented in an integration activity. As presented in Chapter 4, the methodology consists of six processes: analysis, classification, development, testing, deployment and release and quality phases. This methodology is instantiated through a support frame presented in Section 4.3. This framework links the outcomes of the methodology to a support framework where specific technology approaches are evaluated and chosen for a particular integration process.

Goal 3 is achieved through the classification phase of the methodology. In this phase the web applications involved in the integration process are categorised based on their characteristics against the classification scheme introduced in Chapter 2. This determines which data integration process should be implemented. This phase acts as a set of guidelines for the data integrator to use to determine and map out which path to take for their data integration plan. The classification scheme introduced in chapter 2 is used to examine the characteristics and boundaries of the web application and its data sources to categorise which web application type the application belongs to.

Goal 4 is achieved as outlined in Section 6.2 supporting the integrator and system requirements.
Goals 5, 6 and 7 are achieved through GIWeb’s combined methodology and framework as the approach and its evaluation of appropriate technologies phase. The methodology acts as a discipline and set of procedures to follow as part of the integration process with the framework acting as a support tool to guide a data integrator in choosing an integration approach or technology for their integration activity. This means that the approach can be applied to all data integration scenarios as it is appropriate for any integration landscape. It enabled a data integrator to follow an evaluation decision making path in the cases discussed in Chapter 5 for any number of integration technologies and can apply to varying types of integration scenarios.

GIWeb enabled a data integrator in the cases discussed in Chapter 5 to implement existing approaches and those that mature by including these as part of the evaluation process. If the characteristics of a web application show that particular existing approach is appropriate to implement in their integration activity, the data integrator can choose this method. As these existing approaches mature, they may be approach to a wider range of web application types and their characteristics will change. By using the methodology and framework to evaluate appropriate approaches to choose from, the data integrators choices will grow as different current approaches mature to support a wider range of web application characteristics. Figure 6.7 illustrates the current approaches in contrast to Figure 6.8 which shows the matured wider range of approaches.

![Figure 6.7 UML Activity Diagram Current approaches](image-url)
It is this same workflow that enables the data integrator to determine if approaches require reliance on an IT professional to implement and manage the process. If the characteristics of a particular integration technology have a high reliance on a technical expert to manage compared to an alternative approach, the data integrator can make a decision on which approach is more appropriate for their needs and therefore choose the approach that doesn't have reliance on a technical expert. The frameworks survey style questions to determine the different characteristics of integration approaches, specifically addresses the reliance on a technical expert as one of the measures.

Goal 7 is achieved through the multiple stages of GiWeb. The Analysis phase activities are the first step for a data integrator to avoid or resolve compatibility issues in their integration process. These activities as discussed in Chapter 4 are summarised below:

- Collecting user requirements and the data needed to presented to the user
- User classification
- Assigning priorities to each requirement
- Determining data sources
• Determining web applications being integrated
• Conceptual design of the overall data integration architecture between the different web application data sources and web applications being integrated
• Determine technical environments of each web application and data source involved

Collectively they enable the data integrator to properly analyse and map out their integration landscape and design which will help guide the foundations of the integration process and rule out conflicting integration approaches.

The classification phase enables the data integrator to apply the classification scheme to compare the characteristics of different approaches to test their suitability. The classification dimensions as defined in Chapter 2 include the following:

• Data Translation dimension
• Programmable dimension
• Centralised dimension
• Meta-data dimension
• Self-manageable dimension

These dimensions help the data integrator rule out approaches that are not appropriate due to potential compatibility issues.

GIWeb’s framework proposed in Chapter 4 is structured in a survey style with recommendations based on each outcome. The framework presents a series of questions based on determining the characteristics of a web application and expected integration outcome. These questions allow a data integrator such as a web application builder to contrast their web application against the framework criteria and classify it under a specific category of web application. This enabled a data integrator to avoid or resolve compatibility issues in their integration process in the cases discussed in Chapter 5.
6.6 Requirements evaluation

Table 6.11 summarises the requirements of Chapter 3 and how they were met.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Success Criteria</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Integrator Requirements</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1: Accessing the data from the</td>
<td>The process of integrating data sources for the purposes of presenting the data in</td>
<td>GIWeb supports this requirement as outlined in Section 6.2. GIWeb enables a data integrator to choose an appropriate integration technology to use in their integration process by comparing the different technology choices and eliminate selections that would not be suitable for accessing the data from the various sources in a consistent, timely and accurate manner.</td>
</tr>
<tr>
<td>various sources in a consistent,</td>
<td>a web application must provide a means of accessing the data from the various</td>
<td></td>
</tr>
<tr>
<td>timely and accurate manner.</td>
<td>sources in a consistent, timely and accurate manner.</td>
<td></td>
</tr>
<tr>
<td>2: Low reliance on a third party</td>
<td>The ability to implement, manage and control the integration process without the</td>
<td>GIWeb supports this requirement as outlined in Section 6.2. GIWeb enables a data integrator to choose an appropriate integration technology to use in their integration process by comparing the different technology choices and eliminate selections that would not be suitable due to ongoing reliance on a third party or highly skilled technical person.</td>
</tr>
<tr>
<td>or highly skilled technical person</td>
<td>ongoing reliance on a third party or highly skilled technical person</td>
<td></td>
</tr>
<tr>
<td><strong>System Requirements</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1: Design Process</td>
<td>The integration process must describe the architecture being integrated with the</td>
<td>GIWeb supports this requirement as outlined in Section 6.2. The analysis, classification and development phases are an aspect of GIWeb that supports the design process requirement. The cases in chapter 5 demonstrates how these phases enabled the data integrator to map out the right path to follow for their integration process and validated the technical requirements for their integration process.</td>
</tr>
<tr>
<td>2: Heterogeneity</td>
<td>The integration process must support heterogeneous environments, integration from a</td>
<td>GIWeb supports this requirement as outlined in Section 6.2. The data integrator in each of the cases in</td>
</tr>
</tbody>
</table>
consisting of different structures and formats from mainframes to messaging systems. This could include different file formats, access protocols and query languages, different ways of representing and storing the same data and even relationships between data. Chapter 5 was able to determine the characteristics of a particular approach to determine its heterogeneity.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
<th>GIWeb Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>3: Structure and Integrity Support</td>
<td>The integration process must support data quality and data governance. The data translated or transitioned from one source to another must not lose its integrity. Its format may change from one source to the other as part of the integration process but the information contained should be presented the same without any loss or misinterpretation from what as in the original source.</td>
<td>GIWeb supports this requirement as outlined in Section 6.2. Structure, integrity support and performance are a requirement met through the quality phase. In the cases discussed in Chapter 5, this phase enabled the data integrator to examine the speed of the data transformation and presentation between sources. Measuring the ability to handle unpredictable data and testing the ongoing integration of data sources is successful.</td>
</tr>
<tr>
<td>4: Performance</td>
<td>The integration process must ensure the data integration process maintains optimal performance when integrating between a data source to its destination. There should be reduced lag in presentation or reduced loss in quality of data. The framework should have the ability to recover from errors such as source or target systems going down and data quality issues.</td>
<td>GIWeb supports this requirement as outlined in Section 6.2. The test phase and the quality phase enable the data integrator to ensure performance of the data integration process. A performance indicator is that there is no lag in presentation or loss in quality of data. When GIWeb was applied in the cases discussed in Chapter 5, the test phase validated the technical integration method and the flow of the data between data sources.</td>
</tr>
</tbody>
</table>

Table 6.11 Requirements evaluation
6.7 Summary

This chapter evaluated GIWeb for supporting the integration of data sources for use in web applications. It highlighted the various positive and negative effects of applying the approach to specific integration scenarios. It evaluated GIWeb against the Integrator and System requirements detailed in Section 3.5 and the classification scheme proposed in Chapter 2 to demonstrate the applicability of GIWeb for supporting integration of data sources with web applications.

This chapter argued that GIWeb met the requirements described in Section 3.5 evaluating the qualitative outcomes. The classification scheme was used to compare the characteristics of the approach against the classification scheme dimensions to assess the ability to support integration of data sources. It evaluated the qualitative outcomes of GIWeb for improving on current approaches for the integration of data sources for use in web applications. It showed that the approach enabled the data integrator to properly analyse and map out their integration landscape and design to help guide the foundations of the integration process and rule out conflicting integration approaches.

It enabled the data integrator to apply the classification scheme to compare the characteristics of different approaches to test their suitability. The dimensions of the classification scheme helped the data integrator rule out approaches that are not appropriate due to potential compatibility issues. GIWeb is structured in a survey style with recommendations based on each outcome. It was shown that the framework helps the data integrator determine the characteristics of a web application and expected integration outcome. It allows a data integrator such as a web application builder to contrast their web application against the framework criteria and classify it under a specific category of web application. This enables a data integrator to avoid or resolve compatibility issues in their integration process.

A contribution of this chapter was examining the GIWeb’s applicability in being used to evaluate different data integration technologies for web applications. This evaluation was done using several case scenarios. The case scenarios presented evaluated the
challenges faced when integrating using some of the approaches discussed in chapter 2 and then an evaluation of the framework as a solution. These cases highlighted the different data integration environments and the challenges integrating data sources with web applications in these environments when applying some of the approaches discussed in chapter 2 such as data integration models or technical approaches. It also examined the outcomes of how a web developer built a web application applying GIWeb discussed in Section 5.7. It argued beneficial support for the data integration through improving their decision making process and selection of integration technologies for their integration process. GIWeb was shown to help narrow down which integration option to select based on the specific integration process. It enabled the web developer to have the confidence in their decision based on a thorough checking process through GIWeb.
7 Conclusions

7.1 Overview

This chapter presents the conclusions of the thesis. Section 7.2 presents an overview of the thesis. Section 7.3 highlights the major contributions of the research presented in this thesis; this includes the classification and characteristics of different applications for integrating data sources. Major contributions include the evaluation of generic models, management process-based and technology approach groups for integrating data sources for use in web applications and GIWeb for supporting the integration of data sources for use in web applications. Section 7.4 discusses potential future work which could be undertaken in this area and how the work presented here could be applied to other fields. Finally section 7.5 presents some concluding remarks.

7.2 Overview of the thesis

This thesis has described and evaluated a methodology for supporting integration of data sources for use in web applications. Chapter 1 introduced the field of data integration, the characteristics and motivation for supporting a data integrator in their integration process. Web applications rely on data from heterogeneous sources. The data from data sources are often structured differently and are difficult for a web application builder to integrate. Current approaches are not necessarily appropriate in all integration scenarios because of their domain and application specificity. The research presented in this thesis examined solutions for the integration of data sources for use in web applications to support web application builders. It aimed to improve the area of web application development by defining a methodology for supporting the integration of data sources for use in web applications with a supporting framework.

Chapter 2 introduced the field of data integration. It provided an analysis on the background of data integration, the models and technical approaches for data integration
and characteristics of each. It described a classification scheme of model types and technical approaches. It examined the similarities and differences between them to determine their appropriateness for supporting a data integrator in their integration process. It highlighted difficulties associated with data integration with existing management process-based approaches, model and technical approaches. It identified the need to improve on current approaches to support new and emerging models and technologies and be applicable for the maturity and reusability of existing technologies for data integration. It highlighted the need to overcome compatibility issues, high cost to maintain and reliance on a technical expert with existing approaches.

Chapter 3 presented the need for a solution to aid the data integrator in their integration process based on the analysis discussed in Chapter 2. It presented the need for a methodology and framework as the ideal approach. This chapter presented the need for requirements for this solution. It introduced the requirements for supporting data integration with data sources for web applications. It examined how to identify requirements, approaches and validation of requirements. It presented a list of requirements for supporting data integration with data sources for web applications.

Chapter 4 presented the proposed approach GIWeb, a methodology for supporting data integration of web applications and its support framework. The chapter described the methodology used to achieve the objectives of this thesis instantiated through a framework. It proposed a methodology for data integration for web applications and a framework. It described how it can be followed by a data integrator to achieve their data integration outcomes. It described how it supports the data integration process for a broader set of integration goals than the domain specific objectives of current approaches described in Chapter 2. It examined how the methodology bridges the gap between technologies and processes, distinct from generic IT management frameworks and can be extended to a greater range of technologies.

Chapter 5 focused on the use of GIWeb for supporting the integration of data sources for use in web applications. It presented several cases where GIWeb is implemented with a focus on how it supports the integration of data sources for use in web applications. It examined the application of GIWeb to different types of integration environments including the Eclipse development platform and the Zend framework. GIWeb was
applied to the project management framework PRINCE2 where the conceptual management of an integration project was modeled out. The focus on varying types of integration environments examined how GIWeb interacted with the whole development process and the benefits of it. It compared the data integration outcomes for when GIWeb wasn’t applied to the scenario and the outcomes of when it was.

Chapter 5 also presented a demonstration of developing a web application using the GIWeb plugin for the Zend Framework. This chapter presented a scenario where the application needs to receive data and integrate it with an external server. It applied the application of GIWeb as a solution and examined how the web developer would implement this.

Chapter 6 presented an evaluation of GIWeb’s methodology and supporting framework. It discussed how the application of GIWeb supported the requirements discussed in Chapter 3. It evaluated GIWeb against the Integrator and System requirements using the cases from Chapter 5. It evaluated GIWeb against the classification scheme presented in Chapter 2 to demonstrate the applicability of GIWeb for supporting the integration of data sources for use in web applications. It evaluated the qualitative outcomes of applying GIWeb to the cases presented in Chapter 5 and contained an analysis of the quantitative outcomes. It also examined the alternative outcomes of not applying GIWeb to the cases in Chapter 5. The evaluation demonstrated that the application of GIWeb supports the web developer in their integration process.

7.3 Major contributions

Three questions were posed in this thesis which this thesis presented a detailed evaluation to answer, which is summarised as follows:

1: What challenges are currently faced when attempting integration of data sources for use in web applications?

This thesis highlighted how choosing an appropriate data integration approach and technology is an important step in the web development process and that it requires
deep knowledge of many possible approaches and technologies. The consequences of choosing the wrong approach or technology could result in the data integration failing or only achieving a partial success, poor quality data results or incorrectly matched data fields (see Section 2.5). This can lead to unsuccessfully achieving the data integration goal.

2: What characteristics of web applications determine what integration method to use for integrating of data sources?

This thesis demonstrated how the characteristics of web applications help to determine what integration method to use for integrating data sources. Classification is done by categorising competencies examining the common criteria that describe the hierarchy of different application types. The classification was based on five criteria boundaries being Data Translation dimension, Programmable dimension, Centralised dimension, Meta-data dimension and Self-Manageable dimension (see Section 2.3.3).

This has led to the major contribution of this work:

**Classification of different web applications**

The thesis presented an evaluation of the process to follow for identifying the characteristics of a web application. It examined how to use these characteristics to compare them to the characteristics of suitable integration technology approaches to identify the most appropriate technology to implement for an integration process. It showed how the data integrator is able to identify the type of approach they should be looking for and also rule out approaches that will not be appropriate for their type of integration process (see Section 4.3).

This has led to the major contribution of this work:

**Characteristics of web applications determining what integration method to use for integrating of data sources**

A list of requirements was identified, broken up into two groups being integrator requirements and system requirements. The requirements focused on a user who is attempting an integration process would expect to see and system requirements which
are technical outcomes that need to be supported. It showed that these requirements can be used to evaluate the support a data integration framework provides for successful integration with data sources for web applications (see Section 3.5.3).

This has led to the major contribution of this work:

**Identification and validation of requirements for the integration of data sources for use in web applications**

3: Can a high-level methodology be instantiated using a framework to support integration of data sources for web applications?

A major contribution of this thesis was the proposed approach GIWeb (see Sections 4.2 and 4.3) for supporting the integration of data sources for use in web applications. GIWeb was presented as a comprehensive and detailed evaluation of the applicability of a high-level methodology instantiated using a framework to support the integration of data sources for web applications. An examination was discussed on how the methodology improves on the current approaches. It described how this would become a workflow for developing dynamically integrating applications through a support framework. A methodology was presented as an organised, documented set of procedures that can be used as a step by step approach to carrying out the intended integration procedure. It was evaluated for how it helps the data integrator determine goals, motivation and purpose for the integration task and allows an integrator to assess where their application fits and what integration process to follow. It is used to consult and aid the integration of data sources through web applications. The methodology was divided into six phases: Analysis phase, Classification phase, Development phase, Test phase, Deployment and Release phase and Quality phase. The methodology is instantiated with a support framework examining how it acts as a generic framework and guide that a data integrator could follow to achieve their integration goals. An evaluation was examined for this approach for enabling the data integrator such as a web developer to choose the most appropriate integration approach to implement. The framework is broken up into three phases: Identify characteristics of the web application, compare web application characteristics to characteristics of suitable approaches and identify appropriate technology to implement for integration process.
This has led to the major contribution of this work:

**An approach for supporting the integration of data sources for use in web applications**

This thesis has presented a detailed evaluation of the data Integration models to support the data integration process for use in web applications (see Section 2.4). This has shown that models groups such as layer models, common, schema, data mapping and language models all boast strong support for data integration. It showed that many of the model groups are intended to support integration within a specific domain. The evaluation focused on the importance of models in the data integration process supporting the data integrator in their integration process. It showed each category of models has its application and benefits but also demonstrates a common pattern of difficulties associated with data integration. These range from being domain specific, strong technical reliance for maintaining and implementing the model. There are problems associated with time by handling the way data sources change over time and lack of support for new and emerging technologies.

This has led to the major contribution of this work:

**Evaluation of generic models for integrating data sources for use in web applications**

It examined technical approaches to data integration describing the technical methods that support or achieve data integration by means of utilising software (see Section 2.5). A model approach lays down the framework for how a technology approach is implemented. Each technology evaluated demonstrated a range of benefits. This thesis described groups of technical approaches such as Standardisations, Brokers, Extensible Languages, Exposed APIs, Public APIs and Extract Transform Loads. The evaluation showed all boasted strong support for data integration but many of these are intended to support integration within a specific domain or application area.
This has led to the major contribution of this work:

**Evaluation of generic technology approach groups for integrating data sources for use in web applications**

This thesis presented a detailed evaluation of management process-based approaches consisting of software development processes, methodologies and frameworks that can be used to improve integrating data sources for use in web applications (see Section 2.6).

The viability of each group of management process-based approaches was examined. This included the processes of extending existing frameworks, adapting an IT management framework or altering software development frameworks and toolkits. This thesis showed extending existing frameworks or altering software development frameworks may serve as the foundation to build on as a solution for integrating data sources for use in web applications. It showed that project management frameworks were not suitable in all integrating scenarios. Extending existing frameworks showed criticisms where the framework is not a technical one and it involved founding a solution to integrating data sources for use in web applications on a framework not designed specifically to address this technical challenge. When examining IT management frameworks, the body of literature showed factors that can influence each frameworks implementation included being seen as complex, too general and overlapping each other.

This has led to the major contribution of this work:

**Evaluation of generic management process-based approaches for integrating data sources for use in web applications**
7.4 Future work

The research presented for supporting the integration of data sources for use in web applications could be expanded into other technologies. Mobile phone applications, wearable technology applications and virtual reality interfaces are just a few related domains where applications rely on data sources. Developers who create applications in these areas face the same challenges that web developers face when creating web applications. This leads to the need for extending the research presented to these other domains and expanding the use of GIWeb beyond just web applications.

A limitation of the GIWeb approach is tied to the data integrator correcting identifying the characteristics of the web application. If the web developer does this incorrectly, this will lead to an invalid integration technology suitability match from following GIWeb. This leads to an opportunity for further research in the field to expand how a data integrator could better identify the characteristics of web applications.

The expansion of GIWeb could include the classification of different mobile applications determining what integration method to use for integrating of data sources on mobile devices. GIWeb could support a data integrator in the identification and validation of requirements for the integration of data sources for use in mobile applications. It could support developers in the integration of data sources for use in wearable technologies. Virtual reality platforms such as Oculus Rift will see marketplaces of applications connected over the Internet that will require the same challenges arise and the need for the same support for data integrators.
7.5 Closing comments

Commercially the outcome of this research helps support corporations, IT professionals and researchers in their need to integrate with data sources for use in web applications. The current state of art discussed showed organisations are making increasing use of web applications and web-based systems as an integral part of providing a service using the Internet. In order for web applications such as personalised dynamic user content on a website, social media plug-ins or interface mapping software to have maximum use they require the integration of data from multiple sources. This thesis has shown integration of data from multiple sources is problematic and there are a wide variety of integration approaches for a developer to consider. The research presented has a highly commercial appeal for assisting a developer in their integration process overcoming these challenges. The support for the data integrator in selecting the integration approach most appropriate to their individual integration scenario will save both time and improve decision making during the integration process. Increasing productivity through saving time and improved decision making in an integration process could equate to a strong financial saving for commercial companies engaged in data integration objectives. The cost of implementing an ineffective integration outcome could be disastrous and require a complete re-development of the web application or starting the entire integration process from scratch.

There is value for future work outside the scope of web applications. It would be beneficial to implement and evaluate this scope for other technology types as new platforms emerge. As web application development continues to evolve quickly, many new technologies are being introduced for web developers to utilise. There is an ongoing need to consolidate these technologies to aid the web developer. There will always be value in research to assist in consolidating web development and data integration technologies together to make the data integration process more refined as the Internet evolves.
Appendix 1

/* ... */
'modules => array(
    'Application',
    'ZF\Apigility',
    'ZF\Apigility\Provider',
    'AssetManager',
    'ZF\ApiProblem',
    'ZF\MvcAuth',
    'ZF\OAuth2',
    'ZF\Hal',
    'ZF\ContentNegotiation',
    'ZF\ContentValidation',
    'ZF\Rest',
    'ZF\Rpc',
    'ZF\IntegrationPlugin',
    'ZF\Versioning',
    'ZF\DevelopmentMode',
    // any other modules you have...
),
/* ... */
Appendix 2

<extension
id="%oda.data.source.id"
point="org.eclipse.datatools.connectivity.oda.dataSource">
<dataSource
defaultValue="%data.source.name"
driverClass="org.eclipse.birt.report.data.oda.jdbc.dbprofile.impl.Driver"
id="%oda.data.source.id"
odaVersion="3.2"
overrideExplorerFiltering="true"
setThreadContextClassLoader="false">
<properties>
<property
allowsEmptyValueAsNull="false"
canInherit="true"
defaultDisplayName="%datasource.property.databasename"
isEncryptable="false"
name="databaseName"
type="string">
</property>
<property
allowsEmptyValueAsNull="false"
canInherit="true"
defaultDisplayName="%datasource.property.username"
isEncryptable="false"
name="username"
type="string">
</property>
<property
allowsEmptyValueAsNull="false"
canInherit="true"
defaultDisplayName="%datasource.property.password"
isEncryptable="true"
name="password"
type="string">
</property>
<property
allowsEmptyValueAsNull="false"
canInherit="true"
defaultDisplayName="%datasource.property.jdbcURL"
isEncryptable="false"
name="URL"
type="string">
</property>
<property
allowsEmptyValueAsNull="false"
canInherit="true"
defaultDisplayName="%datasource.property.driverClass"
isEncryptable="false"
name="driverClass"
type="string">
</property>
<property
allowsEmptyValueAsNull="false"
canInherit="true"
defaultDisplayName="%datasource.property.jarlist"
isEncryptable="false"
name="jarList"
type="string">
</property>
</properties>

<relationship
   relatedId="org.eclipse.datatools.connectivity.db.category"
type="wrapperOf">
</relationship>

</dataSource>
< dataSet
defaultDisplayName="%data.set.name"
id="org.eclipse.birt.report.data.oda.jdbc.dbprofile.sqbDataSet">
<properties>
<propertyGroup
defaultDisplayName="Query Properties"
name="queryProperties">
<property
type="string"
defaultDisplayName="%dataset.property.parameterMetaData"
canInherit="true"
name="parameterMetaData"/>
</propertyGroup>
<propertyVisibility
   name="parameterMetaData"
   visibility="hide">
</propertyVisibility>
</properties>
< dataTypeMapping
   nativeDataTypeCode="1"
   odaScalarDataType="String"
   nativeDataType="CHARACTER">
</dataTypeMapping>
< dataTypeMapping
   nativeDataTypeCode="4"
   odaScalarDataType="Integer"
   nativeDataType="INTEGER">
</dataTypeMapping>
< dataTypeMapping
   nativeDataTypeCode="8"
   odaScalarDataType="Double"
   nativeDataType="DOUBLE">
</dataTypeMapping>
< dataTypeMapping
   nativeDataTypeCode="6"
   odaScalarDataType="Double"
   nativeDataType="FLOAT">
</dataTypeMapping>
< dataTypeMapping
   nativeDataTypeCode="3"
   odaScalarDataType="Decimal"
nativeDataType="DECIMAL">
</dataTypeMapping>
<dataTypeMapping
nativeDataTypeCode="91"
odaScalarDataType="Date"
nativeDataType="DATE">
</dataTypeMapping>
<dataTypeMapping
nativeDataTypeCode="92"
odaScalarDataType="Time"
nativeDataType="TIME">
</dataTypeMapping>
<dataTypeMapping
nativeDataTypeCode="93"
odaScalarDataType="Timestamp"
nativeDataType="TIMESTAMP">
</dataTypeMapping>
<dataTypeMapping
nativeDataTypeCode="16"
odaScalarDataType="Boolean"
nativeDataType="BOOLEAN">
</dataTypeMapping>
<dataTypeMapping
nativeDataTypeCode="12"
odaScalarDataType="String"
nativeDataType="VARCHAR">
</dataTypeMapping>
<dataTypeMapping
nativeDataTypeCode="5"
odaScalarDataType="Integer"
nativeDataType="SMALLINT">
</dataTypeMapping>
<dataTypeMapping
nativeDataTypeCode="-5"
odaScalarDataType="Integer"
nativeDataType="BIGINT">
<alternativeOdaDataType
odaScalarDataType="String">
</alternativeOdaDataType>
<alternativeOdaDataType
odaScalarDataType="Decimal">
</alternativeOdaDataType>
</dataTypeMapping>
<dataTypeMapping
nativeDataTypeCode="7"
odaScalarDataType="Double"
nativeDataType="REAL">
<alternativeOdaDataType
odaScalarDataType="Decimal">
</alternativeOdaDataType>
</dataTypeMapping>
<dataTypeMapping
nativeDataTypeCode="2"
odaScalarDataType="Decimal"
nativeDataType="NUMERIC">
</dataTypeMapping>
Appendix 3

<html>
<body>
<center>
<table>
<tr>
<td>
<form name=frm method=post action=receive.php>
Customer No: <input type=text name=customerno>
<br><br>
Customer Name: <input type=text name=customername>
<br><br>
Address: <input type=text name=address>
<br><br>
Company Name: <input type=text name=companyname>
<br><br>
<input type=submit name=submit value="Submit">
<br><br>
</form>
</td>
</tr>
</table>
</center>
</body>
</html>
include "db.php";
// include database file

$CustomerNo = $_REQUEST["CustomerNo"];  
$CustomerName = $_REQUEST["CustomerName"];   
$CustomerAddress = $_REQUEST["CustomerAddress"];   
$title = $_REQUEST["CustomerAddress"];   
$status = $_REQUEST["status"];  
$companyId = $_REQUEST["companyId"];    
$Date = $_REQUEST["Date"];   

// instantiate DOM Document
$objXML = new DOMDOCument("1.0"); //writing ROOT element
    $web = $objXML->createElement("web:UpdateRequest");
    $attrWeb = new DOMAttr("xmlns:web",$url);
    $web->appendChild($attrWeb);
    
// Message Header Element
    $msgHeader = $objXML->createElement("MessageHeader");
    $Originator = $objXML->createElement("Originator ",$originator);
    $msgHeader->appendChild($Originator);
    
    $CustomerNo = $objXML->createElement("CustomerNo",$CustomerNo);
    $msgHeader->appendChild($CustomerNo);
    
    $ReturnCode = $objXML->createElement("ReturnCode",$returnCode);
    $msgHeader->appendChild($ReturnCode);
    
    $CustomerName = $objXML->createElement("CustomerName");
    //if the CustomerName is defined, it should be appended to the node
    if ($CustomerName) {
        $web->appendChild($CustomerName);
    }
    
    $CustomerAddress = $objXML->createElement("CustomerAddress");
    if ($CustomerAddress) {
        $web->appendChild($CustomerAddress);
    }
    
    $Title = $objXML->createElement("Title",$title);
    $web->appendChild($Title);
    
    $Status = $objXML->createElement("InducteeStatus",$status);
    $web->appendChild($Status);
    
    $Date = $objXML->createElement("ExpiryDate",$Date);
    $web->appendChild($Date);
    
    $Company = $objXML->createElement("Company");
    $web->appendChild($Company);
    
    $companyId = $objXML->createElement("companyId",$companyId);
    $web->appendChild($companyId);
$CompanyName = $objXML->createElement("CompanyName", $companyName);
$web->appendChild($CompanyName);
$objXML->appendChild($web);
// format xml output
$objXML->formatOutput = true;

// creates unique XML file
$path = "xml";
$fileName = $path . time() . ".xml";
if ($objXML->save($fileName) != FALSE) {
    echo "XML file <a href="$fileName">$fileName</a> generated successfully.";
} else {
    echo "Failed!";
}

$file = "$fileName";
$remote_file = "$fileName";

// set up basic connection
$ftp_server = "externalseverIP";
$conn_id = ftp_connect($ftp_server);

// login with username and password
$ftp_user_name = "FTP-User";
$ftp_user_pass = "Password";

$login_result = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);

// upload a file
if (ftp_put($conn_id, $remote_file, $file, FTP_ASCII)) {
    echo "<br><br>Successfully uploaded $file\n";
} else {
    echo "<br><br>There was a problem while uploading $file\n";
}

// close the connection
ftp_close($conn_id);
?>
References

APA Style


Agichtein, E., Lawrence, S., & Gravano, L. (2001). *Learning search engine specific query transformations for question answering*


Alberts, D. (1996). *The Unintended Consequences of Information Age Technologies*


APIGility. Apigility by Zend Framework.


Deutch, D., Greenshpan, O., & Milo, T. (2010 ). Navigating in Complex Mashed-Up Applications

Dialect, A. “App” voted 2010 word of the year by the American Dialect Society.


ETL. (2012). ETL poll produces unexpected results.


Feiler, J. (2000). *Application servers: powering the web-based enterprise*


Giordano, A. D. (2010). *Data Integration Blueprint and Modeling: Techniques for a Scalable and Sustainable Architecture*


Gujarathi, G. P. (2011). *Parametric CAD/CAE integration using a common data model*


Hohpe, G., & Woolf, B. (2003). *Patterns and best practices for enterprise integration*


Jakl, M. (2005) REST, Citeseer *Representational State Transfer University of Technology Vienna*


Jinan, G., & Guojing, L. (2010). *International Conference on Mechanic Automation and Control Engineering*

Kahn, J. (2006). Infrastructure software standardization is the key to data center standardization. *Enterprise Networks & Servers (Online).*


Mozafari, B., Zeng, K., & Carlo. (2010). Regular Expressions to Nested Words: Unifying Languages and Query Execution for Relational and XML Sequences

Computer Science Department.


Xu, G., & Papageorgiou, L. (2007). *A mixed integer optimisation model for data classification*

