



DSPACE

<https://dspace.org/>

A methodology for evaluating capability, effort and ease of implementation in modular web content management systems

Ramnath, Aveer

2017

<https://hdl.handle.net/10539/24243>

Ramnath, Aveer (2017) A methodology for evaluating capability, effort and ease of implementation in modular web content management systems, University of the Witwatersrand, <<https://hdl.handle.net/10539/24243>>

UNIVERSITY OF THE WITWATERSRAND

**A methodology for evaluating capability,
effort and ease of implementation in
modular web content management
systems**

by

Aveer Ramnath

A dissertation submitted in fulfillment of the requirements for the
degree of Master of Science in Engineering

in the
Faculty of Engineering and the Built Environment
School of Electrical and Information Engineering

August 2017

Declaration of Authorship

I declare that this dissertation is my own, unaided work, other than where specifically acknowledged. It is being submitted for the degree of Master of Science in Engineering at the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other university.

Signed:

Date:

“Every program has (at least) two purposes: the one for which it was written, and another for which it wasn’t.”

Alan J. Perlis

Abstract

Modular web content management systems (WCMS) are widely adopted software platforms that facilitate the creation of web applications through a process of configuration and assembly of add-on modules. Although WCMSs have been used in a variety of application domains (e-commerce, news) no clear guidance as to when it is suitable to use a WCMS could be found. This work proposes a methodology to evaluate the suitability of a WCMS in a particular context. This is done by evaluating the suitability indicators (capability, effort and ease of implementation) for a given WCMS application. The methodology evaluates each indicator per application requirement. Capability is evaluated on a Yes/No basis. Effort is evaluated using effort level, a relative indicator of effort. Effort levels are defined in terms of increasing effort, varying from 0 (feature present in the product) through to 5 (feature requires a custom module to be written). Ease of implementation is evaluated using a qualitative measure (easy, moderate or difficult) of the implementation difficulty. The methodology has been successfully validated through the development and evaluation of a web application for a school within a university faculty. In this instance the WCMS capability was evaluated at 100%, as all requirements could be implemented. The effort level analysis showed 16% of requirements were present by default in the core product, 22% required some configuration of the core product, 32% required a single add-on module to be installed, and 30% required multiple add-on modules to be installed. The ease of implementation analysis showed that 86% of requirements were easy, 7% moderate and 7% difficult. The analysis is presented in order to demonstrate the operation of the methodology. Further data would be needed to extrapolate general trends. With repeated use of the methodology in various contexts, it would be possible to build up a useful reference for those considering the use of a WCMS. In addition, this data would permit analysis of overall strengths and weaknesses of a particular WCMS.

Acknowledgements

I would like to thank Professor Barry Dwolatzky for his patience, wisdom and guidance during this work. I would also like to acknowledge the support of the School of Electrical and Information Engineering in providing me the time and resources to pursue this work. Lastly I would like to thank my wife, Tanisha, for her unwavering support, without which this would not have been possible.

Contents

Declaration of Authorship	i
Abstract	iii
Acknowledgements	iv
List of Figures	ix
List of Tables	x
Abbreviations	xii
1 Introduction	1
1.1 Modular WCMS	1
1.2 Core features of modular WCMSs	2
1.3 Using WCMS as an application framework	2
1.4 Problem statement	3
1.5 Why is this research useful?	3
1.6 Structure of dissertation	4
2 Background	6
2.1 Evolution of web authoring techniques towards web content management systems	6
2.1.1 Hand-authored pages	7
2.1.2 Web applications and databases	7
2.1.3 From web applications to web content management systems	7
2.2 What is a Content Management System?	8
2.3 Examples of WCMS applications	9
2.4 What is content?	9
2.5 WCMS capabilities	10
2.6 Popularity and adoption	10
2.7 The WCMS ecosystem	11
2.8 Community involvement	12
2.9 Constructing a website using a WCMS	12
2.9.1 Setup of core application	13
2.9.2 Customisation of core application	13

2.9.3	Installing add-on modules to extend functionality	14
3	Literature survey	16
3.1	WCMS research	16
3.2	Research on evaluating software	17
3.3	Research on evaluating OTS software	18
3.4	Evaluating capability, effort and ease of implementation in OTS projects .	19
4	Problem statement, research question and approach	20
4.1	Problem statement	20
4.2	Research question	21
4.3	Importance of research	21
4.4	Research approach and expected results	21
5	Development of methodology to evaluate suitability indicators	23
5.1	Evaluating indicators per requirement	23
5.2	Classifying requirements into groups	23
5.3	Evaluating capability	24
5.4	Evaluating effort	25
5.5	Evaluating ease of implementation	28
6	Application specification and requirements grouping	29
6.1	Choice of application	29
6.1.1	Application characteristics	29
6.1.2	Developing the website for a school within a university	30
6.2	Requirements elicitation	30
6.2.1	Application purpose	30
6.3	Requirements listing and classification	31
6.3.1	Presentation requirements	31
6.3.2	Domain requirements	31
6.3.3	Infrastructure requirements	36
6.4	Choosing a WCMS platform	37
7	Application implementation	39
7.1	Implementation of presentation requirements	39
7.1.1	P1: Clean Uniform Resource Locators	39
7.1.2	P2: Dynamic menu	41
7.1.3	P3: External links	42
7.1.4	P4: Standardised footer to appear on all pages	43
7.1.5	P5: General page layout	44
7.1.6	P6: Homepage layout	46
7.1.7	P7: Menu	46
7.1.8	P8: Menu breadcrumbs	47
7.1.9	P9: Theme	47
7.2	Implementation of domain requirements	48
7.3	Implementation of domain entity requirements	49
7.3.1	D1: Course	49
7.3.2	D2: Bursary	49

7.3.3	D3: Facilities	50
7.3.4	D5: News	51
7.3.5	D7: Postgraduate Student	51
7.3.6	D8: Publications	51
7.3.7	D9: Research Group	51
7.3.8	D10: Staff	52
7.3.9	D13: Vacation Work	52
7.3.10	Implementation of domain entity requirements	52
7.3.11	Modules that support domain entities	52
7.3.12	Work flow to implement an entity using CCK family of modules	53
7.3.13	Summary of implementation of domain entity requirements	56
7.4	Implementation of domain non-entity requirements	57
7.4.1	D4: Generic pages	57
7.4.2	D11: User groups and access control	58
7.4.3	D11: User management	58
7.5	Implementation of infrastructure requirements	59
7.5.1	I1: Backup	60
7.5.2	I2: Email	60
7.5.3	I3: Image display	61
7.5.4	I4: Input sanitation	61
7.5.5	I5: Link Checker	62
7.5.6	I6: Logging	62
7.5.7	I7: Popular links	62
7.5.8	I8: Private pages	63
7.5.9	Regular tasks	63
7.5.10	I10: Rich text editors	64
7.5.11	I11: Search	65
7.5.12	I12: Search engine optimisation	65
7.5.13	I13: Site map	66
7.5.14	I14: Site statistics	66
7.5.15	I15: Site structure export	67
8	Application testing, deployment and operation	69
8.1	Approaches to testing WCMS applications	69
8.2	Functional testing	70
8.3	Limited testing needed when compared to bespoke development	70
8.4	Non functional testing	72
8.4.1	Page load times	72
8.4.2	Website load	72
8.4.3	Security	73
8.5	Application deployment	74
8.6	Failures during operation	74
8.6.1	Power failure	74
8.6.2	Corrupt user session table	74
8.6.3	Disk space exhausted	75
8.6.4	Presentation rendering errors	75
8.6.5	Implementation errors	76

8.7	General maintenance	76
9	Results and Discussion	77
9.1	Evaluation of capability	77
9.2	Evaluation of effort level	78
9.2.1	Analysis of overall effort level results	78
9.2.2	Split of core and add-on modules used for implementation	79
9.2.3	Analysis of working with many third party modules	79
9.2.4	Effort levels for presentation requirements	80
9.2.5	Effort levels for domain requirements	82
9.2.6	Effort levels for domain entities	82
9.2.7	Effort levels for domain non entities	83
9.2.8	Effort levels for infrastructure requirements	83
9.3	Evaluation of ease of implementation	84
9.3.1	Overall ease of implementation	84
9.3.2	Ease of implementation for presentation requirements	86
9.3.3	Ease of implementation for domain requirements	87
9.3.4	Ease of implementation for infrastructure requirements	88
9.4	Limitations of study	88
10	Conclusion	90
A	Detailed Application Requirements	93
A.1	Definition of properties for domain entities	93
A.2	List of options for specific properties	94
B	Implementation Details	98
B.1	Modules used for implementing requirements	98
B.2	Listing of fields for domain entity types	98
	References	102

List of Figures

2.1	Initial configuration of Drupal 7 WCMS core application	13
2.2	Bare-bones version following initial setup of Drupal 7	14
2.3	Drupal website module administration page	15
2.4	Drupal website following customisation [31]	15
5.1	Modelling effort level on the basis of effort and module origin	28
6.1	Mockup of web page layout	32
6.2	Example of operations on a staff member domain entity	35
6.3	Domain specific access control for a group of users of school website . . .	36
6.4	CKEditor: a rich-text editor that is used in many WCMSs [54]	37
7.1	Structure of a Uniform Resource Locator (URL)	40
7.2	Dynamic menu that expands as it is navigated	41
7.3	An image indicating an external link	43
7.4	Blocks can be freely placed in a region defined by theme	44
7.5	Layout of general page	45
7.6	Example of a menu breadcrumb showing location in site hierarchy	47
7.7	Display of fields for a Course entity	49
7.8	Listing of Course entities for 4th year of study	50
7.9	Different presentation formats of staff and course entities	53
7.10	Portions of administrative page generated by Content Creation Toolkit .	55
7.11	Layout of course entity using Display Suite Module	55
7.12	Views module control panel	56
9.1	Percentage of requirements that the WCMS platform was capable of re- alising	78
9.2	Percentage of requirements implemented per effort level	79
9.3	Split of core and add-on modules used for implementation	80
9.4	Effort level split for presentation requirements	81
9.5	Effort levels for domain entity requirements	82
9.6	Effort levels for infrastructure requirements	83
9.7	Ease of implementation for overall application	84
9.8	Ease of implementation for presentation requirements	87
9.9	Ease of implementation for domain requirements	88
9.10	Ease of implementation for infrastructure requirements	89

List of Tables

2.1	CMS example websites categorised by application domain [20]	9
2.2	Content types defined per CMS application example[20]	10
2.3	Key features of WCMS platforms [10], [22]	11
2.4	Popular WCMS platforms [26]	12
4.1	Expected results for evaluating capability	22
5.1	Definition of effort levels	27
6.1	Presentation requirements for the school website	32
6.2	Domain requirements for school website	33
6.3	Classifying requirements as domain entities	33
6.4	Properties of a domain entity such as a course	34
6.5	Abridged options for property Schools responsible for course	34
6.6	Infrastructure requirements for school website	37
7.1	Summarised results for presentation requirements	48
7.2	All fields for Course data type	50
7.3	Summarised results for domain entity requirements	57
7.4	Summarised results for domain non-entity requirements	59
7.5	Summarised results for infrastructure requirements	68
8.1	Response times for the ten most popular pages on the application	72
8.2	Errors encountered during live operation over 24 month period	75
A.1	Properties of Bursary	94
A.2	Properties of Course	94
A.3	Properties of News	94
A.4	Properties of Page	94
A.5	Properties of postgraduate	95
A.6	Properties of research group	95
A.7	Properties of staff	95
A.8	Properties of Vacation Work	95
A.9	Options for Schools responsible for course	96
A.10	List of options for staff positions	97
A.11	List of options for programs offering course	97
B.1	Modules used for implementing requirements in Category 1-3	99
B.2	All fields for Course entity	99
B.3	All fields for Bursary entity	99

B.4	All fields for Facilities entity	100
B.5	All fields for News entity	100
B.6	All fields for Postgraduate Student entity	100
B.7	All fields for Publication entity	100
B.8	All fields for Research Group entity	101
B.9	All fields for Staff entity	101
B.10	All fields for Vacation Work entity	101

Abbreviations

API	Application programming interface
CCK	Content Creation Kit (as defined in Drupal)
CMS	Content management system
COTS	Commercial off-the-shelf software
CRUD	Create Update and Delete
CSS	Cascading Style Sheets
HTTP	Hypertext Transfer Protocol
LAMP	Linux, Apache, MYSQL, PHP application stack
OTS	Off-the-shelf software
RSS	Rich Site Summary
SQL	Structured Query Language
URL	Uniform Resource Locator
WCMS	Web Content Management System
WWW	World Wide Web
WYSIWYG	What you see is what you get
XML	Extensible Mark-up Language

Chapter 1

Introduction

Within its short lifespan the world wide web (WWW) has experienced tremendous growth [1]. De Kunder's research indicates that there are billions of pages on the web [2]. A range of technologies to create, publish and manage web content exist; each with varying requirements of technical skill [3, p.4-7]. This study focuses on one particular class of products that are used to create websites, the Web Content Management System (WCMS).

The Oxford Dictionary defines a Content Management System (CMS) as “a system designed to manage the content of a website or other electronic resource that is used collaboratively by a number of people” [4]. W3Tech, a company which tracks the technologies used in the top 10 million websites, reports that 44.3% of these websites are built using a WCMS [5].

1.1 Modular WCMS

This study restricts the focus to modular WCMSs, that is content management systems that are built using reusable software components called modules. The modules are general purpose in nature and through appropriate configuration and combination can be used to produce a desired result. A number of the popular open-source WCMSs (Wordpress, Drupal and Joomla) are built using a modular architecture [6]. Yuventi and Cortez state that the modular architecture permits modules to easily be installed and removed, and thus enables extensibility and flexibility. Furthermore the typical

WCMS is essentially a framework and most functionality is provided through modules [7]. The module concept has been fully embraced by the development community and Wordpress, Drupal and Joomla report the number of developed modules numbering in thousands to tens of thousands as of August 2016 [8] [9] [10]. The WCMS is thus an example of Off-The-Shelf (OTS) Based Software Development [11], that is a software system that is built using third party components.

1.2 Core features of modular WCMSs

The following list summarises the common features of a modular WCMS [3, p.3] [6] [7]:

- Define a site structure and navigation system
- Permit non technical editors to update site content
- Allow for the user interface theme (colours, fonts, layout) to be configured or replaced
- Implement rules to govern the display of content e.g. display the last n posts
- Provide URLs that are accessible and thus are optimised for indexing by search engines
- Control permissions and roles which enable effective site management
- Permit third party developers to modify functionality through the use of modules
- Support assembly of sites by those with little or no programming experience

These features have enabled rapid development of websites and contributed to the widespread adoption of WCMS [7].

1.3 Using WCMS as an application framework

WCMSs are no longer being used exclusively to building content-rich websites, and instead are being extended to deliver feature rich web applications, which contain complex work flows and logic [7]. Examples of such applications are e-commerce sites, collaboration tools, learning management systems and news services [12] [13] [14]. Websites

such as `theeconomist.com`, `store.bbc.com` and `weather.com` are examples of large and complex web applications built on a WCMS platform. In certain instances complete solutions have been created to suit a specific industry. As an example, WooCommerce is an e-commerce solution built on the WordPress platform. `BuiltWith.com` reports that WooCommerce is the world's most popular e-commerce platform and is estimated to be used in approximately 1.7 million e-commerce stores [15].

1.4 Problem statement

Whilst there are numerous examples of using a WCMS as an application platform, it is still generally unclear as to which applications are suited to this approach. In this context suitability is restricted to factors such as whether an application is realisable, whether the effort involved is reasonable, and whether the resultant system can be maintained. An application developer may be tempted to use a WCMS as a basis for building a particular application, but cannot rely on any body of knowledge to help facilitate this decision.

This research aims to provide some insight into the suitability of using a WCMS as an application platform by focusing on three indicators: capability, effort and ease of implementation. Capability considers whether the WCMS can realise the required features for the application. Effort considers the work involved in implementing a given feature. Ease of implementation is a measure of how easy or difficult it is to implement a particular feature. The approach undertaken is to develop a methodology that enables capability, effort and ease of implementation to be evaluated, and then apply that methodology to a specific application.

1.5 Why is this research useful?

As described earlier, WCMSs are extensively used platforms for building websites and web applications. Despite their popularity no research was found which considers what applications are suited to the WCMS platform. By using the proposed methodology it would be possible to evaluate the capability, effort and ease of implementation involved in the development of a variety of applications. This information is useful as it would

assist a developer in deciding whether a WCMS is an appropriate choice for the intended application. It is anticipated that with sufficient data, it would be possible to determine which classes of applications are well-suited to WCMS.

1.6 Structure of dissertation

The dissertation is structured as follows:

- **Chapter 2: Background** - provides an overview of the WCMS ecosystem and discusses its origin, development and current usage.
- **Chapter 3: Literature survey** - surveys research related to evaluating capability, effort and ease of implementation in WCMS and off-the-shelf software systems.
- **Chapter 4: Research question and methodology** - introduces the research question, motivates the importance of the research and describes the methodology used to answer the question.
- **Chapter 5: Development of methodology to evaluate suitability indicators** - analyses the suitability indicators (capability, effort and ease of implementation) in order to determine a method of evaluation.
- **Chapter 6: Application specification and requirements grouping** - chooses an application for development, defines all requirements and classifies them into the requirement groups (presentation, domain and infrastructure).
- **Chapter 7: Application implementation** - implements all requirements and evaluates each of the suitability indicators per requirement.
- **Chapter 8: Application testing, deployment and operation** - discusses the how the application was tested as well as issues encountered during deployment and live operation. Testing covers functional testing, performance and security.
- **Chapter 9: Results and discussion** - summarises the results from the implementation phase and then provides an analysis of these results. Global and specific trends for each requirement group are discussed.

- **Chapter 10: Conclusion** summarises the work done and lists the key findings of the study.

Chapter 2

Background

This chapter serves as an introduction to the Web Content Management System (WCMS) by considering its origin, enabling forces and current state. Additionally the installation and configuration of the WCMS is described, as this will provide the reader with some insight as to how the WCMS is used.

2.1 Evolution of web authoring techniques towards web content management systems

The key enablers of the World Wide Web can be attributed to the development of HTML and Hypertext Transfer Protocol (HTTP) by Tim Berners-Lee [16]. These technologies simplified, unified and standardised the access of documents across a variety of heterogeneous hosts [17, p.64]. Prior to this, accessing resources across a network was highly dependent on the operating system of the host. Web technology significantly reduced the cost and expertise needed to publish content [17, p.98].

Byron et al. [3, p.4-7] and McKeever [18] describe the evolution in web authoring technologies through three distinct paradigms: hand-authored pages, scripting¹ and databases, and content management frameworks. Subsections 2.1.1 - 2.1.3 summarise their findings.

¹The term scripting is used to refer to all web applications that execute code. This work will use the more general term web application, as this refers to both scripted and compiled languages.

2.1.1 Hand-authored pages

Early websites were authored by hand, with the designer required to control both the content and presentation [3, p.4]. Designers required a firm grasp of HTML in order to be able to craft these pages. As a result of this manual approach, these websites were static in nature; meaning that once published the content would not be altered until the next update. Hand-authoring also made it difficult to ensure consistency across a large site [3, p5]. As an example consider the case of updating a standard footer field such as a company slogan. This would require the designer to laboriously open and edit every page individually. The lack of automation in hand authored sites resulted in common problems such as broken web links, inconsistent menus and disparate site layout [18].

2.1.2 Web applications and databases

The proposed solution to the above problems ushered in the age of web applications and databases, wherein web pages were generated by running applications [18]. Frequently these applications persisted data in a database. These applications changed the types of services that could be offered. Automated generation of web pages that leveraged persistent data meant that a variety of applications were now possible. This paradigm enabled the variety of web applications (news, e-commerce, social networking) that are now commonplace [18].

The web application approach helped to ensure consistency across a site and enabled centralised management of site wide updates [3, p.6]. Furthermore it was possible to achieve a separation of concerns regarding data and presentation. By having data stored in the database, it was possible to separate the data management tasks from the HTML presentation issues. Thus the website development team could consist of designers focused on the presentation aspects, and developers focused on the management of application data and logic [3, p.6].

2.1.3 From web applications to web content management systems

The web application technique had proven to be very successful and many bespoke applications were written to satisfy the needs of the customer seeking to manage content

[3, p.7]. Software developers involved in such development recognised commonality between the systems, and progressed towards the development of a generic product that could meet the needs of various customers. A generic product offered the benefit that common infrastructural code (user profile and rights management, site menu structures, backup-restore techniques) could be re-used across customers and systems. System administrators and end users also benefited from a consistent user interface across systems [3, p.7].

The WCMS was born from the recognition that many websites perform a similar function; the delivery of various forms of content (text, images, audio, video) to a community of users [3, p.8]. Consequently work began on a generic product that provided infrastructure to deliver content, thus reducing the efforts to re-implement these features in different applications. Importantly these systems also had the design goal of enabling non-programmer end users the ability to develop and maintain their own web-sites [3, p.18]. Thus by lowering the barrier to entry, it comes as no surprise that the WCMS has been widely embraced [5].

2.2 What is a Content Management System?

The Oxford Dictionary defines a Content Management System (CMS) as “a system designed to manage the content of a website or other electronic resource that is used collaboratively by a number of people” [4]. In its most rudimentary form the content consists of information (text, images, multimedia) that will be created, maintained and delivered to a specific audience. A further useful definition of the CMS is given by Boiko [19]:

“At the highest level, content management is the process behind matching what you have to what they want. You’re an organization with information and functionality of value. They’re a set of definable audiences who want that value.”

The role of the CMS is to co-ordinate the activities of the various role players, and simplify the task of managing the content. Strictly speaking, a CMS is not restricted to being a web based application, but can follow any application paradigm. In this study

the term Web Content Management System (WCMS) is used to refer to a CMS that is deployed on a website. The sections that follow will elaborate on the characteristics of WCMS.

2.3 Examples of WCMS applications

In order to better understand the role of a WCMS, a few examples are given in Table 2.2 [12] [13]. This table lists the application category and websites that have used a WCMS to realise the application. As can be seen, WCMSs have been employed in a variety of domains (news, arts and culture, education). Many of these applications are informational websites, and thus the common thread appears to be the presentation of content.

TABLE 2.1: CMS example websites categorised by application domain [20]

CMS Application Domain	Example websites
News	Popular Science, The Economist
Corporate and Intranet	AOL Corporate
Art and Music	MTV UK, Sony Music
Community portals	Ubuntu Brainstorm
Educational	Harvard, MIT
E-commerce	The Making Spot

2.4 What is content?

Before considering the nature of the WCMS, the question arises as to what exactly is meant by ‘content’? Fraser states that there is some ambiguity with this term [21] but the general consensus is that content refers to information (or data), such as text and images.

In the Content Management Bible [19], Boiko suggests that the concept of content should not be treated lightly and argues that there is a distinction between data and content. Data is considered to be the entities which computer systems manipulate. These entities are devoid of its context and independent meaning. In contrast, content refers to the information which has a specific human context and meaning.

Content also has a format and a structure, which depends on how it is used. Boiko thus warns that developers of CMS should consider the above distinction, as it influences whether the end user finds the content useful and meaningful [19]. Using the above guiding principle, it is possible to identify content types for the applications mentioned in Table 2.1. This table is extended by adding a column, content type, as shown in Table 2.2.

TABLE 2.2: Content types defined per CMS application example[20]

CMS Application Domain	Example websites	Content Types
News	Popular Science, The Economist	Articles, editors, adverts
Corporate and Intranet	AOL Corporate	Documents, policies, software
Art and Music	MTV UK, Sony Music	Artists, musicians, songs
Community portals	Ubuntu Brainstorm	Members, forums, photo galleries
Educational	Harvard, MIT	Courses, staff, timetables, events
E-commerce	The Making Spot	Stock-items, Orders, Promotions

2.5 WCMS capabilities

The key features commonly implemented in WCMSs [10], [22] are listed in Table 2.3. As can be seen WCMSs provide strong support for application infrastructure (caching, persistence, logging), web specific features (menus, links, email), and user management (authentication, internalisation). Many platforms support extensibility through the use of modules.

2.6 Popularity and adoption

The WCMS has been widely embraced; an online catalogue lists 312 different WCMSs as of February 2016 [23]. The web analytics company `builtwith.com` reports that three popular WCMSs, WordPress, Joomla! and Drupal, are actively used by 16 million, 2.6 million and 755 thousand websites respectively [24]. In addition a number of high profile sites such as *Weather.com* (<http://www.weather.com>), *Walt Disney*

TABLE 2.3: Key features of WCMS platforms [10], [22]

Feature	Description
User management	Authentication, authorization, access control
Internationalization	Support for multiple languages
Search	Ability to index and search all content across site
Link management	Administration functions such as link counting and broken link checks
Menu management	Creation of menu links and site navigation
Theming	Ability to change site look through use of templates
Non-functional features	Implements common infrastructural services such as caching, logging and security
Interoperability	Permits data exchange e.g. web services from other applications
Web connectivity	Support for web messaging technologies such as email and news feeds (RSS)
Administration	Standardised interface for site administration
Workflow	Ability to define a workflow for content creation
Media Management	Ability to handle different types of media e.g. audio, video, PDF
Rich Text Editing	Ability to create text that is formatted e.g. font, color, headings
Extensibility	Permits the installation of add-on modules to extend functionality

(<http://thewaltdisneycompany.com>) and *Chicago Sun Times* (suntimes.com) have used a WCMS as the implementation platform for their websites [12] [13]. The activity on forums, blogs, and conferences (WordCamp, Joomla World Conference, DrupalCon) are indicators that WCMS projects have an active and vibrant community of contributors.

2.7 The WCMS ecosystem

There are hundreds of WCMS products available today, and the most popular platforms report weekly downloads in the range of tens to hundreds of thousands [23]. Market research of the open-source modular WCMS solutions [25] reports three dominant platforms: Joomla! [10], Drupal [8], and WordPress [9]. These platforms will be used as

reference systems for the study. These and other popular WCMS platforms are listed in Table 2.4, along with their implementation language.

TABLE 2.4: Popular WCMS platforms [26]

WCMS Platform	Implementation Language
WordPress	PHP
Joomla!	PHP
Drupal	PHP
DotNetNuke	Microsoft .NET
Alfresco	Java
Plone	Python

2.8 Community involvement

The popular open-source WCMSs have significant community involvement [27] [28]. The activities within the community help to share knowledge and provide direction as to future versions of the system. Some of the initiatives are listed below.

- **Web forums** - users interact by creating discussion groups and question and answer forums
- **Online help and documentation** - websites are available that provide help and documentation for the product and add-on modules
- **Tutorials and Books** - a number of online and print publications provide training
- **Conferences** - community organised events discuss new features and future developments

2.9 Constructing a website using a WCMS

A key benefit of a WCMS is that it enables non programmers to create and maintain their own website [3, p.18]. A modular WCMS generally consists of two parts, the core application and add-on modules [29, p.12]. The core application refers to the standard product which consists of a basic feature set to meet the needs of most users. Add-on modules refer to software components that extend or replace the basic functionality provided by the core product [30, p.1].

During this study, a number of websites using the Drupal and WordPress WCMSs have been created. From this work, it is found that there are three principal activities: setup of the core application, customisation of the core application, and setup of appropriate modules (add-ons) which meet specific needs. The following section will elaborate on the above activities, based on the author's experience.

2.9.1 Setup of core application

The WCMS software is initially extracted onto the file system, and then installed into a web-server. Once available in the web-server, the software can then be configured using a standard web browser. Figure 2.1 shows the initial setup screen from Drupal WCMS.

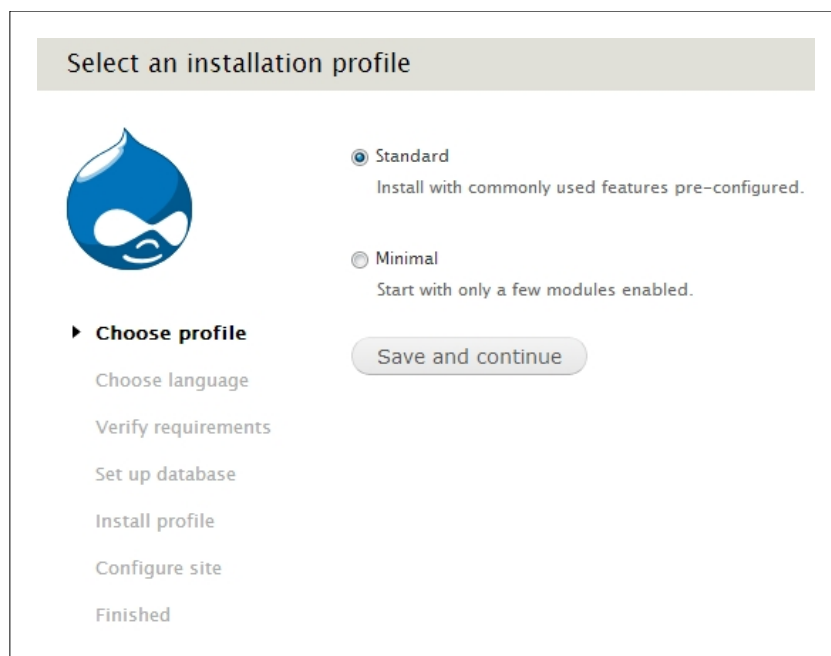


FIGURE 2.1: Initial configuration of Drupal 7 WCMS core application

The site administrator is then able to configure settings such as the site name, URL, database, installation language, and administrator user account. After completing the initial configuration, a bare-bones version of the site is available (see Figure 2.2).

2.9.2 Customisation of core application

Following the initial configuration, the administrator can then customise the site. Some common activities include:

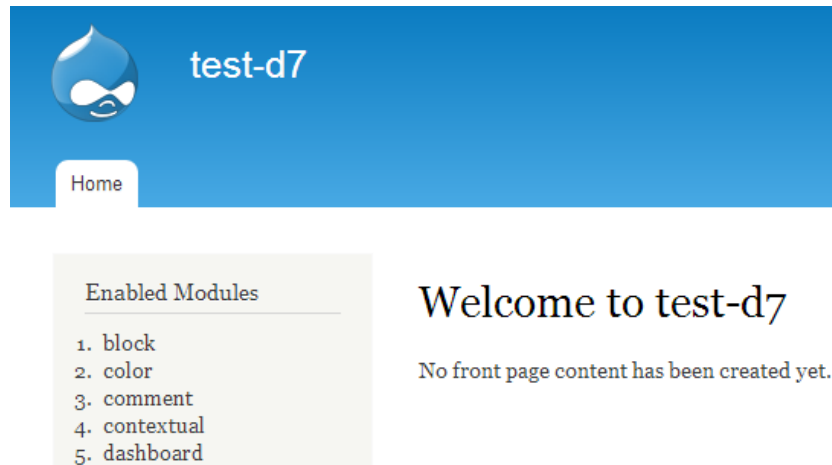


FIGURE 2.2: Bare-bones version following initial setup of Drupal 7

- Choice of site look and feel, usually achieved by selecting a theme.
- Definition of the site structure consisting of linked pages and menus.
- Creation of user accounts for specific roles such as site editors and contributors.

2.9.3 Installing add-on modules to extend functionality

In cases where the core product does not provide a specific feature, the administrator is able to install an add-on module to provide this functionality, should it exist. Most of the popular WCMSs today provide thousands of add-on modules that cater for a variety of uses [8] [10].

Site administrators benefit from the fact that the add-on modules are standardised for a particular WCMS [30, p7]. Consequently all add-on modules conform to a specific structure and interface, allowing them to be plugged-in to the core application. Administrators are thus able to use a common graphical interface to manage the installation and configuration of modules. This greatly reduces the complexity of working with third-party software. A screenshot of the module administration page is given in Figure 2.3.

An example of a fully configured site (as created in this study) is given in Figure 2.4.

This chapter described the evolution of web sites from hand-crafted pages to bespoke web applications and then to common platforms for building web applications. It provided an overview of the WCMS ecosystem and described the basic mechanics of how the

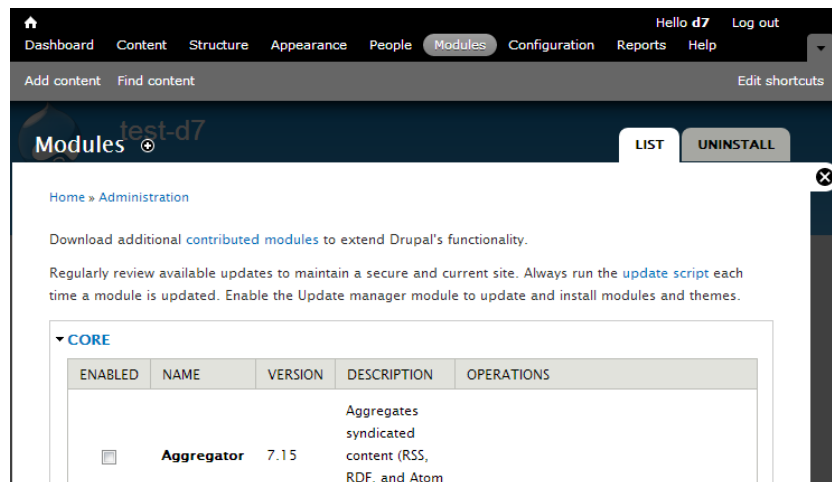


FIGURE 2.3: Drupal website module administration page



FIGURE 2.4: Drupal website following customisation [31]

WCMS is used, as this is essential background to understand the remainder of the work. The following chapter will survey the literature for studies that consider capability, effort and ease of implementation in software systems.

Chapter 3

Literature survey

This chapter presents the results of a literature survey into capability, effort and ease of implementation for WCMS platforms. No research was found that evaluated the above indicators for WCMS platforms in a generic manner. Instead many researchers addressed the above indicators within the context of specific case studies or product comparisons [26], [32]–[34]. The literature survey thus presents a summary of the common research themes related to WCMSs.

Thereafter it was necessary to look further afield to understand existing techniques applicable to evaluating capability, effort and ease of implementation. Off-the-shelf software shares many similarities to the WCMS application assembly approach. Consequently the survey considered research into how off-the-shelf software evaluates the indicators.

3.1 WCMS research

Despite there being a significant amount of work related to WCMS, no research was identified that specifically addressed the capability, effort and ease of implementation in a generic manner. It was found that much of the existing research did touch on the above issues within the context of a specific example. As an example a case study on the use of a WCMS to build an online community [32] would discuss these issues within this context. Similarly a comparison between two WCMSs (Drupal and WordPress) would implicitly address the above issues [33]. Consequently the survey initially looked more broadly at WCMS research. The research areas can be categorised as shown below.

- Implementation of a specific application or system (online communities [32], libraries [34], and research institution websites [35]).
- Broad use of WCMSs within a specific industry (education [36], bioinformatics [37])
- Architecture of specific implementations (service-oriented architectures [38], data-driven systems [7])
- Comparison of capabilities between platforms [26] [33]
- Non-functional software considerations (performance [39], security [40])

Aktunc et al. have considered the use of WCMSs as a rapid prototyping tool for digital enterprises [41]. Their study found that the WCMS approach has provided several advantages to bespoke applications: strong support for collaboration from non-technical users, an established publishing process, and content security. They have also concluded that if carefully designed, the WCMS can be used beyond the prototype stage as a fully fledged application. This work does indicate that WCMSs may be a suitable application platform under certain circumstances.

3.2 Research on evaluating software

As indicated earlier, no research was found that evaluated the capabilities, effort and ease of implementation for a WCMS platform. The WCMS is an example of an off-the-shelf product (OTS), that is a product that delivers a given set of functionality that can be used in various contexts. Thus it would be reasonable to expect that methods which are used to evaluate OTS can be applied to a WCMS. OTS Based Software Development [11] refers to the development of software through the integration of various third party applications. This is similar to the use of third party modules in a WCMS, and thus will also be considered.

3.3 Research on evaluating OTS software

Brown and Wallnau have studied how technology evaluations are conducted for OTS software [42]. Their findings are that most evaluations are done in an ad-hoc manner and consist of some of the following approaches:

- Gather subjective opinions and experiences through engaging with other practitioners
- Engage with vendors to obtain objective data on product capabilities
- Compare functionality of a given product against user requirements
- Conduct focused experiments to mitigate high-risk areas
- Conduct a pilot project to verify feasibility

Jadhav and Sonar have reviewed 64 research papers on the topic of selecting and evaluating software packages [43]. Their conclusions are listed below.

- Software evaluation is a multi-criteria decision making problem.
- Various techniques (feature analysis, weighted average sum, expert system) exist to help the decision maker rank and choose the best alternative.
- The most important activities are to identify criteria for evaluation, assign weights and perform ranking.
- There are no common criteria that are universally used for evaluation.

Ayala et al. have undertaken a study of evaluation techniques used by practitioners [11]. The researchers noted that there is no common practice as to how evaluations are performed. In some cases decisions are made very informally, whilst others apply more rigorous approaches. They have found very little consistency of approach, even within the same organisation.

3.4 Evaluating capability, effort and ease of implementation in OTS projects

Morisio et al. studied 15 off-the-shelf (OTS) projects that were undertaken by NASA [44]. The researchers concluded that the OTS process differed significantly from a traditional software development project. Areas which required special attention in OTS projects were requirements definition, high level design, integration, and testing. Projects reported spending significant effort on integration, and in many cases integration consumed the most effort.

In addition it was recommended that OTS projects also be flexible with requirements as in some cases it may be difficult or impossible to implement some requirements. They regarded it as a trade-off for the reduced costs associated with using an OTS product. From this it is clear that the effort involved in implementing a feature needs to be carefully considered. Even if a feature is possible, it may prove very difficult to implement.

This chapter presented a survey of literature related to the topics of capability, effort and ease of implementation related to WCMSs. It was found that despite significant WCMS research being available, none addressed these topics directly. Instead most research focused on individual case reports of building systems using a WCMS, or comparisons between various WCMS platforms. Lastly a survey of how capability, effort and ease of implementation are evaluated for OTS systems is presented, as these are similar in nature to a WCMS.

Chapter 4

Problem statement, research question and approach

This chapter presents the problem statement, namely that insufficient references to help a developer evaluate whether a WCMS is a suitable platform for a given application exist. Following from this a research question is defined, which is to ask if a methodology can be defined and utilised to evaluate suitability indicators. The importance of the research is presented, followed by a discussion on the research approach and expected results.

4.1 Problem statement

The application developer today is faced with the option of whether to assemble an application using a WCMS platform or to use more traditional approaches such as using general purpose programming platforms. The option of assembling an application from pre-existing components offers benefits such as reduced costs and time-to-market, but also carries the risk that some functionality may be difficult or impossible to implement [44]. At present a developer cannot reference the literature to determine whether building a WCMS is a suitable choice for building a particular application.

4.2 Research question

This research aims to develop and test a methodology that evaluates the suitability of using a WCMS in a given context. Suitability is a broad term and thus it is necessary to limit this to three indicators: capability, effort and ease of implementation.

Following from this, the research question is framed below.

Can a methodology be developed and tested that would evaluate:

1. capability to realise a given application using a WCMS?
2. effort involved in implementing the application?
3. the ease of implementing the application?

4.3 Importance of research

If a number of WCMS implementers apply the methodology, it would be possible to build a database of their experience. This data would be useful to a developer considering using the WCMS within a specific context. As an example, if there are sufficient data points showing a number of developers' experience in implementing a conference application using a WCMS, this would be useful to a new implementer. The database would lead to increased confidence in using WCMS within specific contexts, and consequently more parties would benefit from the efficiencies of using a product compared to bespoke software. Some of these efficiencies are reduced costs, shorter time to market, and fewer defects [44].

4.4 Research approach and expected results

In order to answer the questions posed above, the following research approach will be used.

1. Develop a methodology to evaluate the suitability indicators (capability, effort and ease-of implementation)

2. Define a means of evaluating each suitability indicator
3. Select an application to be used for implementation and evaluation
4. Define the requirements applicable to the chosen application
5. Evaluate the suitability indicators for each requirement
6. Report on overall results and trends
7. Comment on the use of the methodology

The expected results from the evaluation of each indicator are shown in Table 4.1. The specific means of evaluating each indicator will be defined later in the work, thus the tables show a generic measure. Following on from these results, cumulative trends can be investigated and discussed.

TABLE 4.1: Expected results for evaluating capability

Requirement	Capability	Effort	Ease of implementation
Requirement R1	Yes	Small	Difficult
Requirement R2	No	Medium	Easy
Requirement R3	Yes	Large	Easy

This chapter has defined the research problem, namely that there is insufficient information in the literature to support the decision to use a WCMS within a particular context. The research question posed is to investigate if a methodology can be developed and tested which would permit the evaluation of the suitability indicators (capability, effort and ease of implementation). The importance of the research is discussed. The chapter closes with a description of the research approach and expected results.

Chapter 5

Development of methodology to evaluate suitability indicators

This chapter develops the methodology to evaluate the suitability indicators: capability, effort and ease of implementation. The application is decomposed into requirements as each indicator is evaluated on a per requirement basis. Additionally requirements are classified into groups in order to investigate trends that may arise.

5.1 Evaluating indicators per requirement

Each application will be decomposed into its requirements, for the purpose of fine grained analysis. The suitability indicators will be evaluated on a per requirement basis. For the purpose of validating the methodology, this study will define an application, specify its requirements, and then evaluate all suitability indicators.

5.2 Classifying requirements into groups

Many software architectural styles have recognised the need to partition a system in order to achieve a separation of concerns [45, p.19]. The three tier architecture [46, p.244] [47, p.68-72], divides the system into front, middle and back tiers. The front tier considers client or user aspects such as presentation. Typically the front tier contains the graphical user interface of the system. The middle tier contains the application or

business logic. It is in charge of processing requests from the front tier, and processing data from the back tier. The back tier is commonly regarded as the infrastructure tier. It is responsible for persistent storage of application data. Frequently the back tier consists of a database.

It is useful to apply similar partitioning to the system requirements within the problem domain. This separation may lead to insights as to the type of customisation that is predominant for each requirement group. When considering the WCMS capabilities and architecture, as discussed in Section 2.5, it is clear that the same partitioning can be applied to the application requirements.

Infrastructural services are those that are generic in nature and widely applicable to a number of applications [47, p.70]. Examples of such services are security, persistence, and logging. Domain specific refers to requirements that are only applicable to the specific problem domain [47, p.69]. For a conference application domain specific concepts would be speakers, delegates, and talks. Presentation is responsible for displaying information to the user and for processing user input [47, p.70].

Certain requirements cannot be clearly classified into a single group, and in this case, the dominant characteristic should be used as a guide. Thus the classification is not definitive but should be used more as a guideline. The classification into presentation, domain and infrastructural requirement groups will be used for the application developed in this study.

5.3 Evaluating capability

Capability refers to the ability of a WCMS to fully realise a requirement. The evaluation is binary; either the WCMS can realise the requirement or not. Thus the classification for this indicator is either Yes or No. In the event that an existing module cannot be found to satisfy a requirement, it is possible for a developer to write a bespoke module to implement it. In this case the capability is limited by the platform's ability to support a requirement.

5.4 Evaluating effort

Software effort is usually measured in time (staff hours) [48, p.366], but this measure is typically used for accounting purposes. It is difficult to infer an objective measure of work done from a time measure, as individuals work at different rates. Consequently using time as a measure of effort would provide little utility to a developer, as there are too many underlying factors which would influence this measurement.

Programmers sometimes measure or estimate effort by considering the lines of code (LOC) written for a given task [49, p.70] [50]. This approach has also been criticised, since developers produce code at different rates. In addition, a task can be implemented in various ways, with each implementation differing in the number of lines of code needed.

Following on from this, it is clear that it is difficult to measure effort on an absolute scale. It is proposed that a relative measure be defined for the WCMS context. This can be explained through an example. Developers Mary and Alice are working on a project with two tasks, A and B. Using an absolute measure such as time, it may be that developer Mary takes 10 minutes for task A and 30 minutes for task B. Developer Alice may take 20 minutes for task A and 90 minutes for task B. Thus reporting the actual time taken may not be useful, as it is closely related to each developer's way of working.

A relative measure attempts to measure a common characteristic between the two tasks and then compare them to each other. As an example, consider counting the number of arithmetical operations for each task. Applying this technique to the above-mentioned example, consider that task A has 20 arithmetical operations and task B has 50 arithmetical operations. This relative measure indicates that task B requires more effort than A. Function points are an example of such a relative sizing measure [51]. Following on from this, the relative measure called effort level was defined.

When considering the structure of modular WCMS applications it is apparent that there are a number of different ways of utilising the platform (see Section 2.9). Users are able to use functionality that is built into the core product, install add-on modules or themes to extend or alter capability, or author their own modules or themes. The level of effort and expertise needed for each mode of operation may vary significantly; activating a core

feature may only require the setting of an individual switch within the application, whilst authoring a module may require significant time, effort and mastery to accomplish.

A second distinguishing factor to consider is the origin of the module. A module may reside within the product core, as an external third party module, or as a customised module that is authored specifically for a project. The first two types are products since they are used within a variety of projects, whereas the latter is project specific. Whilst this distinction may not be significant to the application constructor, it is nevertheless important to consider. Modules that are add-on are developed by third parties and thus have greater difficulty integrating into a cohesive product [44]. Thus it is worthwhile to distinguish add-on modules from core modules. Secondly the real strength of the WCMS platform is in its ability to be extended and altered with third party modules [8] [9]. Thus it is useful to evaluate this category of modules independently.

The two dimensions mentioned above, effort involved and origin of module, are used as a basis for the effort levels proposed in Table 5.1. It is proposed that as the level increases, so too would the expected effort involved. It is also expected that the origin of the module would influence the effort involved, and thus a core module would require less effort compared to a third party add-on module.

Level 0 represents features that are present within the product core, and require no explicit configuration by the application constructor. Level 1 refers to features that are present within the core of the application but require some degree of configuration for correct operation. Both Level 0 and 1 refer to features that are found in the product core.

Level 2 represents features that are implemented through the use of a single add-on module or theme, whilst Level 3 requires multiple add-on modules. Thus Level 2 and 3 refer to the use of third party add-on modules. It is worthwhile to distinguish between a single add-on module (Level 2) and multiple modules (Level 3), since the integration of a number of independent modules to realise a particular requirement is expected to be more challenging [44]. Thus Level 3 is considered more difficult and is assigned a higher level.

Level 4 and 5 represent project specific modules, that is modules that are bespoke and developed solely for use in an individual project. Level 4 refers to the development of

TABLE 5.1: Definition of effort levels

Effort level	Name	Description
0	Present by default	Feature present in the product core and active by default. Does not require any effort.
1	Configuration of core	Feature present in the product core. Requires configuration of single or multiple parameters.
2	Installation of single add-on module or theme	Feature implemented in a single add-on module. This class also includes the installation of themes which affect layout of application. Usually requires configuration and customisation.
3	Installation of multiple add-on modules	Feature requires multiple add-on modules to be installed. Usually requires configuration and customisation.
4	Coding of theme	Feature requires that a theme be coded. This includes coding from scratch or modification of an existing theme to meet needs. Only required when available themes are not sufficient.
5	Coding of an add-on module	Feature requires that a module be coded. This includes coding from scratch or modification of an existing module to meet needs. Only required when available modules are not sufficient.

a custom theme, whilst Level 5 represents the development of a custom module. It is proposed that theme development is somewhat easier than module development for the following reasons:

- The theming system uses standardised HTML and CSS elements and thus requires no specific knowledge of the WCMS.
- Theming does not involve logical flow and control, but is declarative rather than imperative in programmatic style.
- It has minimal interaction to existing modules and interfaces, when compared to a custom module.

Figure 5.1 shows the effort levels plotted on the dimensions of effort involved and origin of module. The horizontal axis represents effort involved and shows that effort increases from configuration, through to installation and finally coding. The vertical axis shows the origin of the module from the product core, to add-on modules and finally project specific modules. This graph motivates the definition of the effort levels as described earlier.

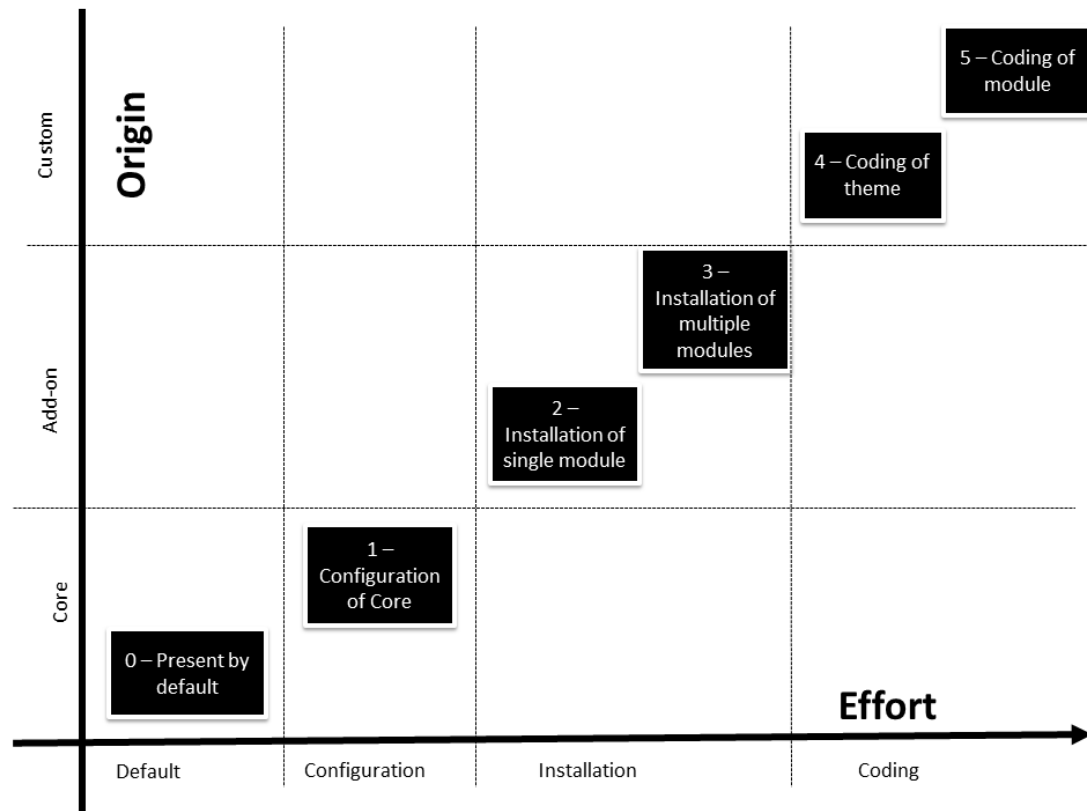


FIGURE 5.1: Modelling effort level on the basis of effort and module origin

5.5 Evaluating ease of implementation

For the implementation of each requirement it is possible to rate how easy or difficult the implementation was. The evaluation must be justified by making reference to particular issues that were encountered during the implementation. This indicator is also useful as it would reveal particular deficiencies or weaknesses in the platform or associated modules.

This chapter presented the need to decompose the application into requirements and requirement groups for the purpose of evaluating the suitability indicators. Thereafter each suitability indicator (capability, effort and ease of implementation) is analysed to find an appropriate means of evaluation.

Chapter 6

Application specification and requirements grouping

This chapter discusses the choice of the application to be developed as well as the specification for the application. In addition to listing the application specification, the requirements are categorised into groups (presentation, domain and infrastructure). Lastly the choice of a suitable WCMS platform for the purposes of the evaluation is discussed.

6.1 Choice of application

6.1.1 Application characteristics

The application must be suited to being developed on a WCMS platform, that is the application should sufficiently exercise the capabilities of the WCMS platform as discussed in Table [2.3](#).

Furthermore it should fit into one of the chosen domains as listed in Table [2.2](#). This would reduce the likelihood that the application cannot be implemented. The question of using a WCMS for a new domain might be useful for future work, but is not suited for this study. Lastly, it would be beneficial if the application could be deployed in a real-world setting. This would prevent the application from being merely a prototype that is never tested in real world conditions.

6.1.2 Developing the website for a school within a university

During the study the opportunity was presented to re-develop the website for the School of Electrical and Information Engineering at the University of the Witwatersrand [31]. This proved to be a suitable candidate since it met the characteristics described above: it was a real-world application, it provided a sufficiently large scope and was within a domain used for WCMS applications.

6.2 Requirements elicitation

The application requirements were defined by following a requirements elicitation process with the various stakeholders. This included academic staff, administrative staff, students, and marketing personnel. The requirements were then categorised into one of three groups: presentation, domain, and infrastructure. The motivation for these groupings was discussed in the previous chapter (see Section 5.2). For the purposes of brevity, details of the requirements gathering process are not presented.

6.2.1 Application purpose

The purpose of the application is to provide an informational portal for the School of Electrical and Information Engineering. There are a number of different stakeholders of this information:

- Prospective students
- Current students
- Alumni
- Staff
- Industry partners and sponsors
- Academic community
- General public

The site would contain information relevant to these various stakeholders. This would include information such as programs on offer, courses, staff members, facilities, events, and news. In addition, such a large site would need to be authored by a number of contributors. Thus the system should implement granular access to portions of the site. Besides the particular domain requirements, it should implement the standard non-functional requirements (quality attributes) of web applications such as availability, scalability and security.

6.3 Requirements listing and classification

Requirements that were gathered were then sorted into the groups of presentation, domain and infrastructure. The results of the categorisation are presented below. The requirements shown here are in an abridged form. Full details can be found in [Appendix A](#).

6.3.1 Presentation requirements

These requirements described elements of the application layout. Factors that influence the presentation are aesthetics, ease-of-use and navigability. A summary of these requirements are given in [Table 6.1](#). Each of the requirements is numbered with the prefix P, which indicates it belongs to the presentation group. A mockup of the graphical user interface is shown in [Figure 6.1](#).

6.3.2 Domain requirements

These are the requirements that are specific to the chosen domain, which in this instance is the academic website of the school. The summarised requirements are listed in [Table 6.2](#), numbered with a D prefix to indicate it is a domain requirement.

These requirements can be subdivided into two classes. The first is referred to as domain entities - those that represent a collection of entities or objects in the domain. For the school web application domain, typical entities are courses, venues, publications, and staff members. Entities have the characteristic that they form a collection. The second class is all other objects that do not exhibit this structure. For the school web application

TABLE 6.1: Presentation requirements for the school website

No	Requirement	Description and basic characteristics
P1	Clean urls	Ensure all pages are addressable with a commonly used format URL e.g. courses/undergrad/1styear
P2	Dynamic menu	Required a menu system that did not occupy too much of real estate on the screen. Envisaged that it would be expandable and dynamic
P3	External links	Visual icon to indicate that links lead to external sites
P4	Footer	Standardised footer to appear on all pages
P5	General page layout	Fixed layout for pages, popular links pane, news item pane
P6	Homepage layout	Specified format for homepage, scrolling banner linking to important content, links to news items, site should be easy to navigate
P7	Menu	Menu system linking to all content across site.
P8	Menu bread-crumb	Provide a breadcrumb type menu which indicates to the user where in the site page hierarchy they reside
P9	Theme	Ability to specify the site presentation theme (colours, font, logo)

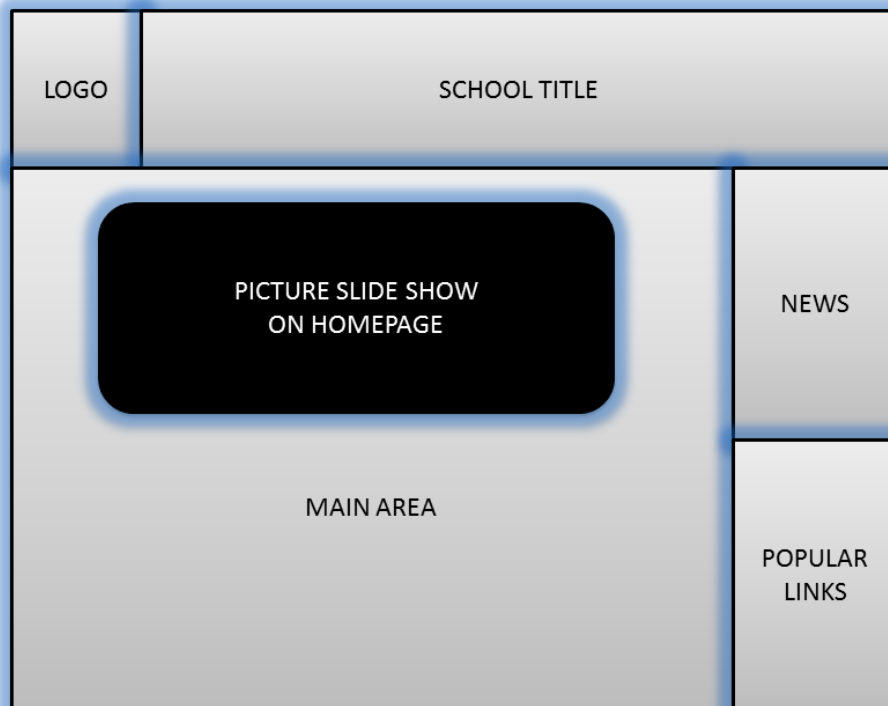


FIGURE 6.1: Mockup of web page layout

TABLE 6.2: Domain requirements for school website

No	Requirement	Description and basic characteristics
D1	Course	Course related information (course description, lecturers)
D2	Bursary	Bursaries and scholarships available to students (companies, application forms)
D3	Facilities	Facilities available in the school (laboratories, lecture rooms)
D4	Generic Pages	Generic information pages describing various functions within the site (location, maps, contact details, history, alumni, community activity, student groups)
D5	News	News stories related to activities within school
D6	Poll	Provide means to poll site visitors on a variety of topics
D7	Postgraduate Student	Biographical information on postgraduate members: (name, email, telephone number, office, research interests)
D8	Publications	Research publications in the school: (conferences, journals, books, abstracts)
D9	Research Groups	Description of work in each research group: (focus, facilities, team members, publications)
D10	Staff	Biographical information on staff members: (name, email, telephone number, office, research interests)
D11	User groups and access control	Allow restricted access to portions of site based on user group
D12	User management	Allow users to create and manage accounts
D13	Vacation Work	Information on vacation work offered to students: (companies, contact details)

this would be pages containing contact information and history. Table 6.3 shows this classification applied to the domain requirements.

TABLE 6.3: Classifying requirements as domain entities

Domain entities	D1, D2, D3, D6, D7, D8, D9, D10, D13
Domain non-entities	D4, D5, D11, D12

Some requirements describe domain entities, that is models of real objects found in the domain that have a unique identity [52]. When considering each entity, it is necessary to define the properties of the entity. As an example, some properties of a staff member would be name, email address and telephone number. Table 6.4 shows the properties for a course.

Each property has a specified data type such as text, integer, or boolean. A full listing of the data types for all the requirements is given in Appendix A. For some properties, additional constraints can be specified. As an example, for the property ‘School responsible for the course’, the choice is constrained to a list of options, as shown

TABLE 6.4: Properties of a domain entity such as a course

Course			
Property	Data Type	Constraint	
Course code	Text	None	
Course name	Text	None	
School responsible for course	Text	Select from a fixed list	
Course co-ordinator	Link	Staff member entity	
Undergraduate course	Boolean	None	
Year of study	Integer	None	
Course brief and outline	Document	PDF format	
Course image	Image	JPG, PNG format	
Programs offering course	Checkbox	None	
Syllabus	Text	Paragraph	

in Table 6.5. Appendix A provides a full listing of options for the applicable fields in the application.

TABLE 6.5: Abridged options for property `Schools responsible for course`

Schools responsible for Course
Accountancy
Anatomical Science
Animal, Plant & Environmental Sciences
Architecture & Planning
Biological and Life Sciences
Chemical and Metallurgical Engineering
Chemistry
Civil & Environmental Engineering

In other cases it is necessary to constrain a field according to a specific format. As an example, an email is a text field but requires that it must contain only one @ symbol and cannot contain any white spaces or illegal characters.

Domain entities are persisted in a database. As such, the common persistent data operations [53, p.92] are applicable. These are summarised below:

- **Create** an entity by specifying its properties
- **Read** an entity
- **Update** an entity by modifying one or more properties
- **Delete** an entity

- **Search** for a collection of entities by specifying a particular property
- **List** the collection of entities of a particular type
- **Filter** a collection of entities by specifying a constraint

As an example consider a university course, which is a domain entity. A course can be created, read, updated and deleted, which are the standard CRUD operations [53, p.92]. A course can be searched for by name or other property. It is possible to list all courses that have been created, as well as filter courses according to a constraint e.g. find all courses that are offered in first year of study. Figure 6.2 shows a similar example for a staff member.

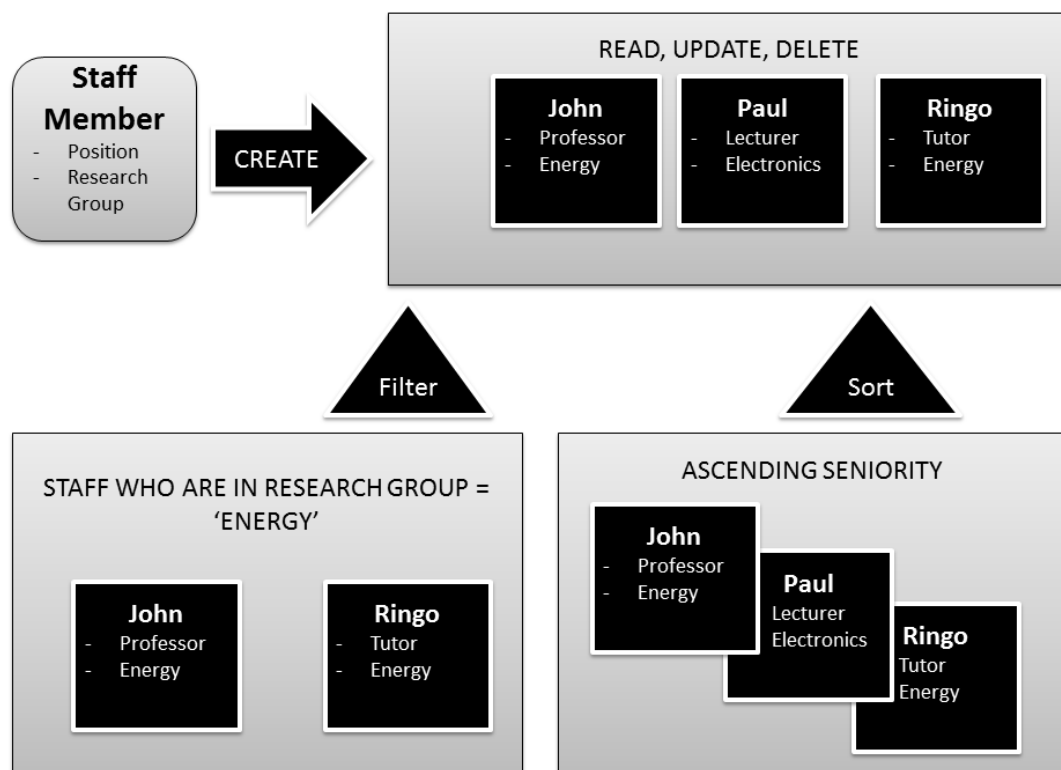


FIGURE 6.2: Example of operations on a staff member domain entity

This class refers to domain requirements that are not considered entities. This usually includes content pages that describe single instances e.g. a description of the history of the school, or a listing of contact details. In addition some domain requirements are not content related, but describe structural issues concerning the application. In Table 6.2 requirements regarding user management and access control are listed as domain non-entities.

User management and access control refers to limiting the access of a group of users to being able to read or modify a particular set of content. In the context of the school web application, an example could be to only allow news items to be edited by a news editor role, but not by general members of staff. Another example is to limit a group of pages to only be visible to staff members.

User management and access control can be considered an infrastructural requirement since this is a common feature required in many domains. However the roles, such as staff member or news editor, are defined by the domain and this is the reason for classifying this feature as being a domain requirement rather than an infrastructural one. Figure 6.3 illustrates this concept by considering the access control differences for an admin user, student and staff member. The student is only able to read public pages, whilst the admin user and staff member are able to read public and private pages.

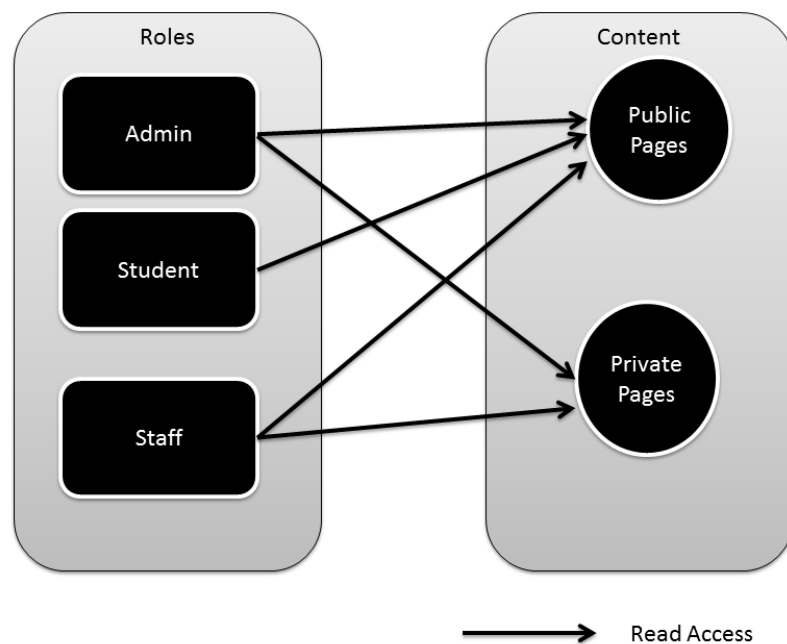


FIGURE 6.3: Domain specific access control for a group of users of school website

6.3.3 Infrastructure requirements

Infrastructure requirements refer to those that are general purpose in nature and usually occur in many different domains. Table 6.6 lists the requirements for the school website. Figure 6.4 shows CKEditor [54], a rich text What-You-See-Is-What-You-Get

(WYSIWYG) editor that is used in many WCMS applications. It allows users to format their text entry as well as include images, video and hyperlinks.

TABLE 6.6: Infrastructure requirements for school website

No	Requirement	Description and basic characteristics
I1	Backup	Archive of site data which can be used in the event of failure
I2	Email	Ability for the application to generate emails
I3	Images	Support for image display
I4	Input sanitation	Prevent malicious user input
I5	Link checker	Check site to determine if a particular URL is invalid e.g. points to a page which does not exist
I6	Logging	Generation of log files to be used for diagnostics
I7	Popular links	Tracking of popular links accessed on the site
I8	Private pages	Restrict pages to be visible to a group of users
I9	Regular tasks	Ability to run regular maintenance tasks
I10	Rich text editors	Allow contributors to create text content using a rich text editor which supports tasks like creating headings or hyperlinks
I11	Search	Enable site wide search and indexing
I12	Search engine optimisation	Ability to tailor page titles and meta-data so that it can be indexed by search engines
I13	Site map	Provide a listing of all pages within the site
I14	Site statistics	Tracking of site statistics (page hits, popular referrers)
I15	Site structure export	Ability to export site structure to search engines

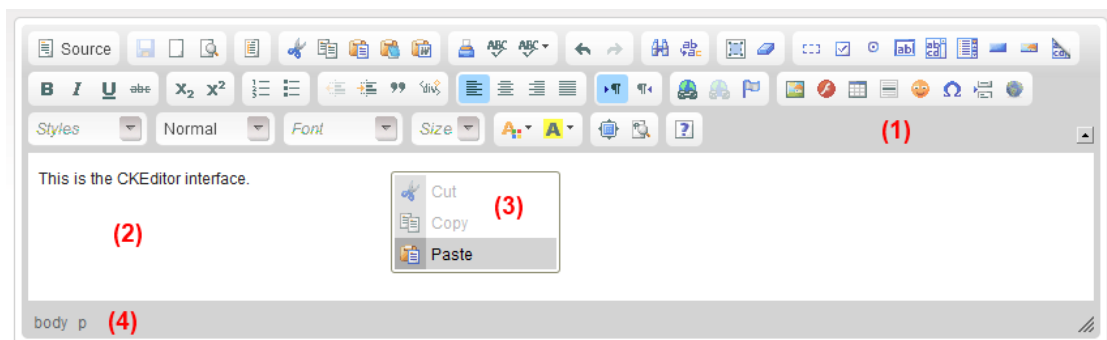


FIGURE 6.4: CKEditor: a rich-text editor that is used in many WCMSs [54]

6.4 Choosing a WCMS platform

In order to facilitate the research aims, it is necessary to choose a platform that is extensible, flexible and capable. The search was restricted to the leading open-source platforms. Bonfield et al. compared the leading platforms Plone, Joomla! and Drupal. Their study concluded that Plone and Drupal were more powerful compared to Joomla

[6]. Drupal had the added advantage of a simpler installation and lower operational costs compared to Plone. In another comparison of Wordpress, Joomla! and Drupal, the study concluded that Drupal is most closely resembles a development platform compared to the others. As such it was regarded as the most extensible platform and is suited to coders [33].

As this research considers using a WCMS platform to implement a custom application, it is prudent to choose a platform that is considered to be the most powerful and extensible. Consequently Drupal was chosen as the WCMS platform for the study.

This chapter presented all the requirements for the application, classified into three groups: presentation, domain and infrastructure. The chapter concludes with a discussion on the choice of WCMS platform, with Drupal emerging as the preferred choice.

Chapter 7

Application implementation

This chapter discusses the implementation of the school website based on the requirements defined in Chapter 6. The implementation is discussed for each requirement group, namely presentation, domain and infrastructure requirements. For each requirement, a general description is given, followed by a discussion of how it was implemented. Further details of the implementation is provided in Appendix B.

7.1 Implementation of presentation requirements

7.1.1 P1: Clean Uniform Resource Locators

A Uniform Resource Locator (URL) is the address used on the World Wide Web to specify the location of a particular resource. A resource could be any object that is capable of being transported over the Internet. Typical resources are HTML web pages, formatted files, images or video. The URL indicates the location of the resource by specifying the domain and server on which the resource resides, as well as the relative path of that resource on the server. This is illustrated in Figure 7.1.

The concept of a path to a file originates from the early web which consisted of static HTML pages [3, p.4]. These pages were arranged in folders and thus the location was specified as a relative path from the root folder of the server. With the advent of server side applications, which generated pages on demand, many web languages abandoned this approach. As an example, PHP adopted the approach of specifying a single target

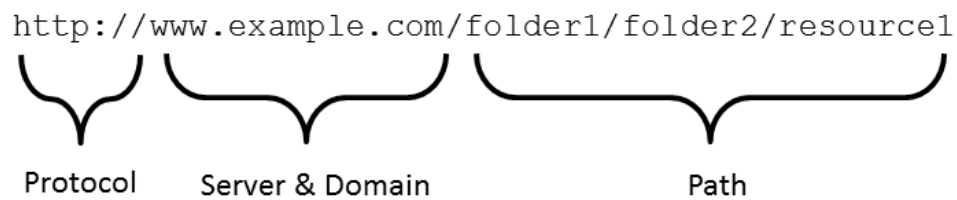


FIGURE 7.1: Structure of a Uniform Resource Locator (URL)

URL which accepted arguments. The arguments, formally referred to as the query string, were able to specify the resource that was required. An example of a URL to access the facilities page is given below:

```
http://www.someschool.ac.za?q=facilities
```

A problem with this format is that it does not keep with the earlier convention that Internet users were accustomed to. Users would like to access individual resources within a site by specifying its path explicitly. The earlier format also had the benefits that it displayed a hierarchical arrangement of resources and thus was easier for a human to understand. Furthermore the traditional format allowed for easier indexing by search engines. Consequently it was required that the school website conform to the traditional format, which are commonly referred to as clean URLs. Examples of clean URLs for the school website are given below.

```
http://www.someschool.ac.za/staff
http://www.someschool.ac.za/staff/joe.smith
http://www.someschool.ac.za/courses/firstyear
http://www.someschool.ac.za/courses/firstyear/ELEN1000
```

The implementation of clean URLs was straight-forward as this was a feature that was present in the product core. The feature was activated by configuring a single parameter in the administration console, and worked as expected. Thus the results of classification are:

- **Capability:** Yes
- **Effort level:** 1 (Configuration of Core)
- **Ease of implementation:** Easy

7.1.2 P2: Dynamic menu

This requirement specified that the menu system used to navigate within the site be dynamically expandable. This was needed since the site consisted of a large number of pages and thus a conventional menu would occupy too much space on the page. A dynamic menu would solve this problem by only expanding when the user selected a particular option. For web applications this is typically solved by using a menu that is implemented using Dynamic HTML (DHTML) and Javascript. Figure 7.2 shows an example of a dynamic menu that expands as a user navigates through it.

In addition the dynamic menu must not conflict with the other presentation requirements specified, and thus it should conform with the overall presentation structure of the site. The type of menu that is used should closely match the chosen theme for the application, and thus this requirement relates to requirements P5 and P6.

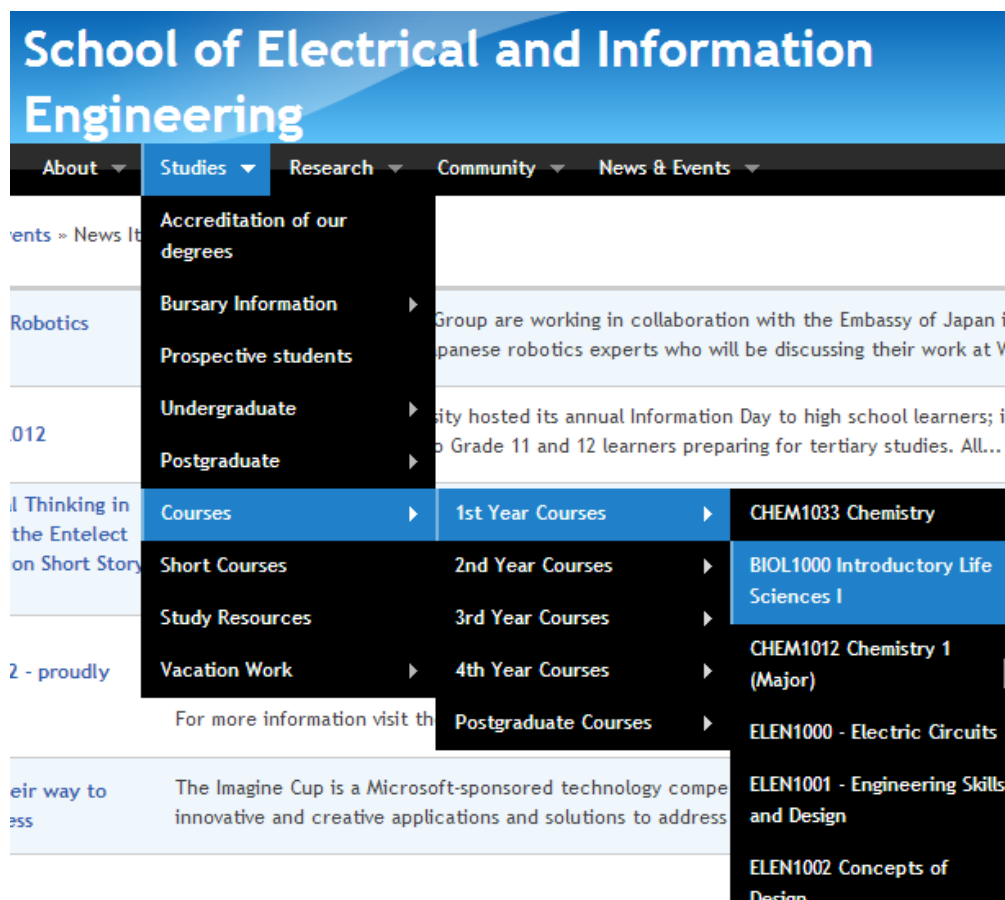


FIGURE 7.2: Dynamic menu that expands as it is navigated

The standard menu that is provided with Drupal core does not provide dynamic functionality. As such it was necessary to find an add-on module that was capable of meeting

this requirement. The open nature of the platform meant number of alternative menus that provide a similar functionality exist. The final implementation required a single module and thus the Effort level is Level 2.

The menu module integrates closely with the existing site theme, and this resulted in many modules being incompatible with the chosen site theme. This meant that a number of dynamic menu modules were tested, and also required that the theme be changed to facilitate compatibility. When problems occurred it was necessary to rely on the documentation of the module and support forums to resolve.

A further issue to consider was the ongoing support for the specific module. In some instances it was noted that the module development was dormant. Using such a module would be unwise since it was unlikely that bug corrections would be forthcoming.

For the implementation of the menu, four dynamic menu modules were tested (see Section B.1 for details). This process consumed a significant amount of time as it required that the theme be altered in an effort to be compatible with the module. For these particular modules, the implementation was difficult as many errors were found. A significant amount of time was spent researching the errors on forums as well as trial and error efforts to resolve faults. Consequently the ease of use can be considered as hard for this particular instance. Finally the **JQuery Menu** module was chosen.

- **Capability:** Yes
- **Effort level:** 2 (Installation of single add-on module or theme)
- **Ease of implementation:** Hard

7.1.3 P3: External links

A typical website contains internal and external links. Internal links are those links that point to pages and content within the site, and external links are links to other sites. Frequently users would like to know if a link is internal or external, since a user may not want to navigate away from a particular site. A common means of achieving this is to append an image next to the link, which indicates that it is an external link. An example of this is given in Figure 7.3.



FIGURE 7.3: An image indicating an external link

This requirement was fulfilled by using a single add-on module, namely **External Links**. The choice was simple since there were no other competing modules and the configuration was easy.

- **Capability:** Yes
- **Effort level:** 2 (Installation of single add-on module or theme)
- **Ease of implementation:** Easy

7.1.4 P4: Standardised footer to appear on all pages

It was required that the footer appearing on all pages be centrally defined and managed. This was to ensure a consistent look and feel across the website. The footer would contain links to a contact page, a site map, central university website, Facebook, and LinkedIn, with the latter two requiring an image to be displayed.

The ability to have a piece of content centrally defined is a commonly required feature in web sites and thus unsurprisingly it is available in the core application. This requirement was implemented using a Drupal block, which is free floating area that can be placed in a specific part of a page. The block can contain various types of content e.g. text and images, and thus was capable of meeting the application requirements.

The area where a block can be placed is determined by the theme in use. Figure 7.4 shows a theme which consists of a top, bottom, left, right, and centre regions. Importantly, the WCMS architecture permits a separation of content from its layout, thus enabling the footer to be defined generically and then positioned using the block and theme system.

The footer was implemented using functionality defined within the core of the application. As such it is classified as Level 1. In addition the implementation was simple and thus the ease of implementation is easy.

- **Capability:** Yes

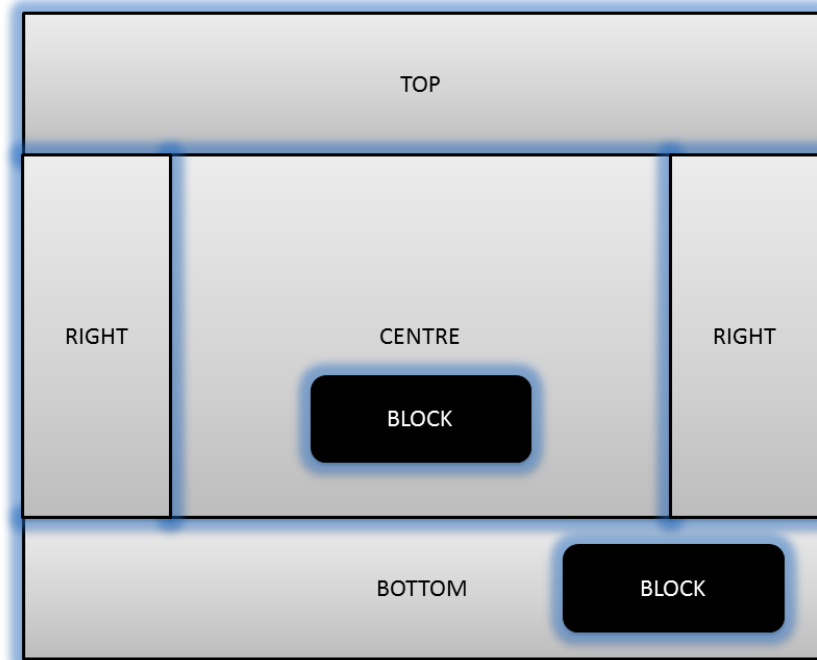


FIGURE 7.4: Blocks can be freely placed in a region defined by theme

- **Effort level:** 1 (Configuration of core)
- **Ease of implementation:** Easy

7.1.5 P5: General page layout

The general page layout referred to features that were present in all pages on the site. Figure 7.5 is a schematic of the expected layout. The general page was specified to have a number of characteristics:

1. The page should display a list of popular links and news items.
2. It should have a site logo and site name.
3. All content should fit a standard screen width.

In order to implement the above, it was necessary to find an appropriate theme. Drupal base distribution is delivered with a number of built-in themes that cater for various look and feel layouts. In addition, a number of third party themes are available for use. The benefit of using a built-in theme is that it is generally more widely used and

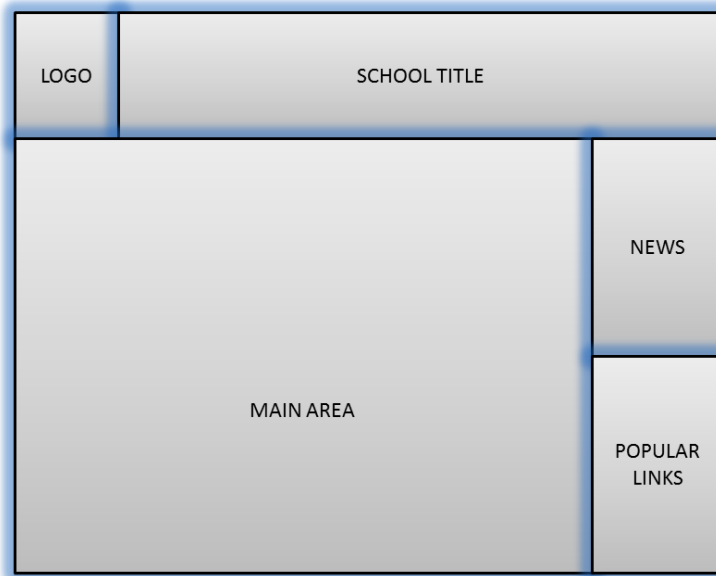


FIGURE 7.5: Layout of general page

thus enjoys a greater degree of support. This also would imply that it would have fewer compatibility issues with other modules.

The theme finally chosen for implementation was **Pixture Reloaded**, which is part of the Drupal base distribution. Before settling on this theme, a number of other themes were tested, but the above theme seemed to provide the best trade-off amongst all the requirements.

The display of popular links and news lists was achieved using the block system as described in Section 7.1.4. It should be noted that this section only discusses the presentation of the content. The implementation of the domain concepts of news items and popular links will be covered in Section 7.2.

This requirement was classified as Level 1 effort since it was achieved using built-in themes and functions, although for other users it could be a Level 2 since an add-on theme may be preferred. The ease of implementation is rated as moderate since a considerable amount of time was needed to test various themes for suitability.

- **Capability:** Yes
- **Effort level:** 1 (Configuration of core)
- **Ease of implementation:** Moderate

7.1.6 P6: Homepage layout

The homepage layout required a few additional features as compared to the general page layout. It was required that the home page contain a scrolling banner which displayed images. When a user clicked on a particular image, it should link to a particular page.

This feature is not available in the product core, and thus required an add-on module. The challenge posed by this feature is that it required integration with an existing theme. In addition, it had to be decided if it was worthwhile to use a theme that already included this feature, or to find a stand-alone module that could be used across multiple themes. Following the testing of various themes and modules it was decided to use the `ddblock` add-on module.

The ease of implementation is rated as moderate since it required some degree of trial and error to determine the correct module to implement as well as to obtain the correct configuration of the chosen module.

- **Capability:** Yes
- **Effort level:** 2 (Installation of single add-on module or theme)
- **Ease of implementation:** Moderate

7.1.7 P7: Menu

It was required to have a hierarchical menu that provides links to all content. It was required to be flexible enough to permit various parts of the menu to point to the same resource, as this would have facilitated easier browsing of the site.

The menu system is part of the Drupal core. It does not require any specific activation as it enabled by default. Thus it is classified as Level 0 with an ease of implementation classified as easy.

- **Capability:** Yes
- **Effort level:** 0 (Present by default)
- **Ease of implementation:** Easy

7.1.8 P8: Menu breadcrumbs

Users of a website often lose track of where they are in the site hierarchy. A breadcrumb is a commonly used technique to help users locate their position. The breadcrumb is text header that appears at the top of each page and indicates where the current page is located within the site hierarchy (see Figure 7.6).

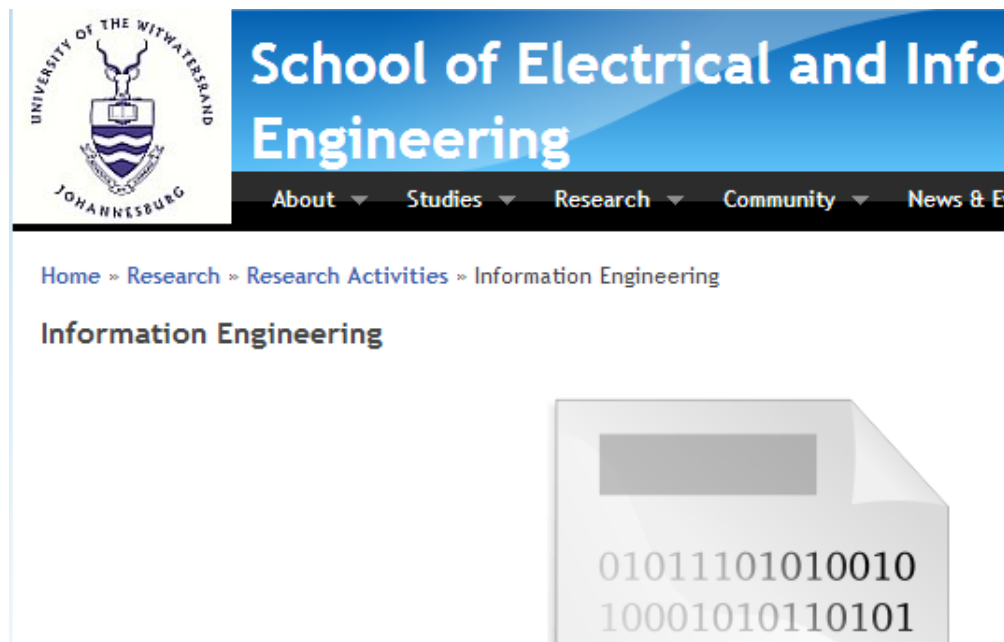


FIGURE 7.6: Example of a menu breadcrumb showing location in site hierarchy

This feature is not present in the core but was available as an add-on module. The implementation was straight-forward and was supported by good user documentation.

- **Capability:** Yes
- **Effort level:** 2 (Installation of single add-on module or theme)
- **Ease of implementation:** Easy

7.1.9 P9: Theme

It was required that the site theme be configurable in terms of font, colours and logo. Following the comparison of various themes available it was apparent that features are not common between them. Whilst most themes offer configuration of logos, many do not support alteration of fonts or colours. The eventual theme chosen was **Pixt**

Reloaded as it offered the best trade-off for all the presentation requirements specified. It also allowed colour customisation, as specified by this requirement. The implementation was straight-forward and only required activation and configuration of the theme.

- **Capability:** Yes
- **Effort level:** 1 (Configuration of core)
- **Ease of implementation:** Moderate

This section discussed the implementation of all the presentation requirements. Table 7.1 summarises the results of the classification according to effort level and ease of implementation.

TABLE 7.1: Summarised results for presentation requirements

No	Requirement	Effort level	Ease of Implementation
P1	Clean urls	1	Easy
P2	Dynamic menu	2	Hard
P3	External links	2	Easy
P4	Footer	1	Easy
P5	General page layout	1	Moderate
P6	Homepage layout	2	Moderate
P7	Menu	0	Easy
P8	Menu breadcrumb	2	Easy
P9	Theme	1	Moderate

7.2 Implementation of domain requirements

The analysis given in Section 6.3.2 indicates that domain requirements can be classified into two groups, domain entities and domain non-entities. Domain entities represent a model of collections that are specific to the domain of the application. Domain non-entities represent all the remaining requirements. The above classification is useful as it turns out that the implementation of domain entities within Drupal share many characteristics. Consequently the description of the implementation of this group can be done collectively.

7.3 Implementation of domain entity requirements

Domain entities share many common attributes. Each of the entities require support for the create, update and delete operations (CRUD). Thus there should be appropriate user interfaces that allow a user to perform these operations. Each entity would consist of fields, which represent the data structure of the entity. As an example, a course entity would consist of course name, course code, and year of study. Each of the fields can be a particular data type and subject to specific constraints. As an example, the year of study is an integer value and is constrained to values between 1 and 5. The listing of all fields for each entity is provided in Section B.2 of Appendix B.

7.3.1 D1: Course

This entity is used to provide details on all courses offered to students. Table 7.2 lists all fields contained within a course entity and indicates the data-types, constraints, and mandatory status. Since there are large number of courses on offer, ease of use dictates that courses should be grouped according to year of study. In addition, certain courses are only applicable to a particular program e.g. Telecommunications Engineering. Thus it must be possible to filter the courses per program of study. Figure 7.7 shows a screenshot of a course listing whilst Figure 7.8 shows a listing of courses filtered for a particular year of study.

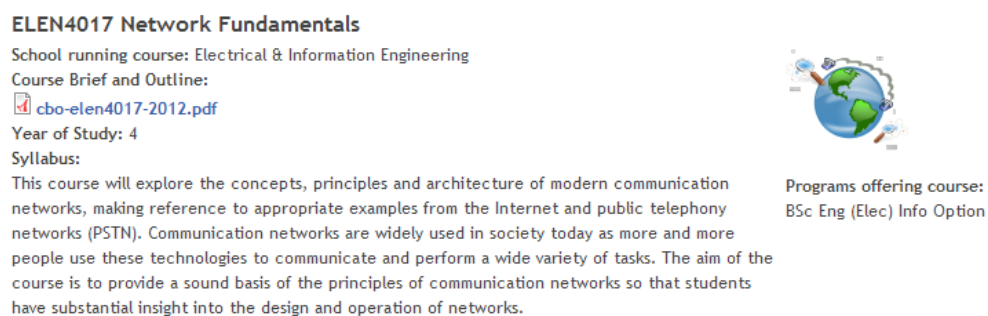


FIGURE 7.7: Display of fields for a Course entity

7.3.2 D2: Bursary

The site should contain a listing of bursaries on offer to students. Each bursary would contain fields such as the contact details of the person administering the program, an

TABLE 7.2: All fields for Course data type

Field Name	Data Type	Input format or Constraint	Mandatory (M) or Optional (O)
Course Code	Simple Text	Fixed format of 4 letters and four digits	M
Course Title	Simple Text	None	M
School Responsible for Course	List	Single choice	M
Course-coordinator	List	Single choice	M
Is Undergraduate Course	Boolean	None	M
Year of Study	Integer	1-5	M
Course Brief and Outline	Document	Allowed formats	M
Programs offering Course	List	Multiple choice	M
Display Image	Image	Allowed formats	M
Syllabus	Multi-line Text	None	M
Course website	URL	None	O

4th Year courses

	Course Code and Name	Offered to
	ELEN4000 Electrical Engineering Design II	BSc Eng (Elec) BSc Eng (Elec) Info Option
	ELEN4001 High Frequency Techniques	BSc Eng (Elec)
	ELEN4003 High Voltage Engineering	BSc Eng (Elec)

FIGURE 7.8: Listing of Course entities for 4th year of study

application form, and a link to a website. Furthermore it should be possible to display a sortable list of all bursaries available.

7.3.3 D3: Facilities

It is required to provide information regarding all the facilities within the school. Typical facilities include laboratories, libraries and work-rooms. For each facility, it is necessary to describe its location, opening hours and amenities, as well as provide a photo gallery.

7.3.4 D5: News

This entity is used to describe news items within the school. The news item consists of a teaser, headline image and general description. The latest news items are also displayed on the homepage.

7.3.5 D7: Postgraduate Student

The postgraduate student entity contains biographical information such as contact information, research group, and research interests. There should be a centralised listing of all postgraduate students as well as the ability to filter them according to specified criteria.

7.3.6 D8: Publications

Publications refer to all scholarly articles that have been published within the school. Typical fields within a publication are publication type (journal, conference, magazine), publication date, research group, and authors. For certain types of publications additional information is needed; for instance a conference listing should specify the venue.

The publications listing is accessed by a variety of users (academics, research group leaders, students). As such, the publication section must support selective filtering and querying. As an example, an academic may require a listing of his/her publications only, a research leader may require a list of all publications for the particular research group, and an administrative officer may require all publications within a particular year. Consequently the publication section should be easily adaptable to permit the variety of queries that may be required.

7.3.7 D9: Research Group

A Research Group refers to a collective of researchers that share similar research interests. The group would contain fields listing the staff and student members within the group, a description of the group's activities, and links to the group's facilities.

7.3.8 D10: Staff

The staff entity contains information like position, research interests, publications and contact details. It is required to query staff members by research group or position. The staff entity must support linking to various other content types such as courses, publications and research groups.

7.3.9 D13: Vacation Work

This entity lists information regarding vacation work opportunities on offer to students. Typical fields would include name and contact information of companies offering the work, description of the type of work and pre-requisites for suitable candidates.

7.3.10 Implementation of domain entity requirements

When comparing the requirements for the domain entities listed above, it is apparent that they share a number of common characteristics. Each entity is required to support the create, read, update and delete requirements. It must also be possible to query the collection of entities for those meeting specific criteria, and then display the returned subset. Furthermore the collections of entities need to be displayed in various forms.

Whilst the entities share common characteristics, their filtering, formatting and display can differ significantly. As an example, the listing of all staff members would be in a tabular format consisting of six columns: a photograph, name, position, telephone, email and interests, whereas the listing of courses is shown in a grid format and displays course name and graphic. This difference is shown in Figure 7.9.

7.3.11 Modules that support domain entities

In recognition of both the similarities and differences amongst domain entities, it was necessary to search for modules that could provide the needed functionality. Following on the analysis given in the previous section, it is apparent that the particular fields and structure of the domain entity is really specific to this individual application. Thus it would be unwise to expect that a pre-built module could be found that would fit exactly to the requirements defined above.



FIGURE 7.9: Different presentation formats of staff and course entities

Drupal developers recognised the variability in domain requirements and thus developed the concept of the Content Creation Kit (CCK) and complementary modules [3], a generic means of defining domain specific content. The rationale behind the CCK family of modules was that it is flexible enough to support the creation of arbitrary content types (entities), whilst providing the plumbing that supports the CRUD operations, search and display. Thus a site constructor would be able to create a custom entity such as a staff member, but would not need to implement any code to persist the entity. Furthermore, it would be possible to query the collection of the entity, and display them in a particular view.

7.3.12 Work flow to implement an entity using CCK family of modules

A significant benefit of the CCK family of modules is that it permits the creation and management of custom entities purely through administration. A user will define an entity (referred to as a content type in Drupal terminology), by defining the fields that

are applicable. For each field, the user will specify its primitive data type (boolean, integer, URL, text field) and the constraints on the data type. Constraints could be limits such as a value between 1 and 10 or a fixed options list such male or female. Using this input data, the CCK modules will then perform a number of tasks:

- **Storage** - Create database schema to persist entities
- **Administrate entity** - Create graphical user interfaces for user to create, update and delete entities
- **View entity** - Provide for pages that display the entity in a user customised format
- **View collection of entities** - Provide pages that show a collection of entities
- **Query for collection of entities** - Permit the definition of custom queries, sorts and filters to display a collection of entities

Following the definition of the entity, CCK will create the database schema to persist the data. Modules such as `Node Import` permit users to import and export data from the CCK database.

CCK will create graphical user interfaces (GUIs) that permit the creation and management of instances of the entity. For a `Course` entity consisting of fields `Course-Image` and `Programs-offering-course`, this will mean that a form will be created whereby a user can create a new course. Consider that field `Programs-offering-course` permits the selection of multiple options from a fixed list, and field `Course-Image` allows for a chosen image to be uploaded. Based on these constraints, the form generated, as shown in Figure 7.10, would permit the appropriate administration of the entity.

When attempting to view an entity, CCK provides a standardised layout of all fields in a list form. In many instances a customised layout is required and thus a module that permitted more sophisticated layout was found. The `DisplaySuite` module was found to be suitable and was used to layout the display of entities. Figure 7.11 shows a layout of a course, defined in `DisplaySuite`, which has the image placed on the right hand side, and other information presented on the left hand side.

Create Course

Course Code and Title: *

Simple Text field

Programs offering course: *

- ☐ BSc Eng (Elec)
- ☐ BSc Eng (Elec) Info Option
- ☐ BSc EngSci (Biomedical)
- ☐ BSc Applied Computing
- ☐ BSc EngSci (Digital Arts)

Multiple Options

Course Image:

Choose File No file chosen Upload

Maximum Filesize: 20 MB

Allowed Extensions: png gif jpg jpeg

Image upload

FIGURE 7.10: Portions of administrative page generated by Content Creation Toolkit

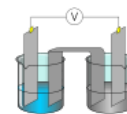
CHEM1033- Chemistry

School running course: Chemistry

Year of Study: 1

Syllabus:

- This unit is designed to expose students not intending to proceed to a second qualifying unit in chemistry to the principles and practical considerations that underlie chemical phenomena in engineering, industrial and mining processes. Topics covered include metals and alloys, the thermodynamics and kinetics of corrosion and other redox reactions, the chemistry of the atmosphere/environment, alternative energy sources, explosives, acids and bases, aqueous solutions and materials. Six laboratory experiments accompany the lectures and illustrate practical aspects of the topic material.



Programs offering course:
 BSc Eng (Elec)
 BSc Eng (Elec) Info Option

FIGURE 7.11: Layout of course entity using Display Suite Module

Frequently it is required to provide an index page that lists a collection of entities. The **Views** module caters for this by providing the capability to define these collections. It is based on a query format where the site constructor can define the fields to be displayed, the sort order, and the number of items per page. An example of two such collections are shown in Figure 7.9.

In many instances it is required to only display a subset of entities. The **Views** module also provides the facility. Take for example the case of displaying all courses within the first year of study, and sorting the courses in alphabetical order of the course title. The above query is accomplished by administration of the control panel shown in Figure 7.12.



FIGURE 7.12: Views module control panel

7.3.13 Summary of implementation of domain entity requirements

The implementation of all domain entities were realised by using the CCK family of modules. This includes modules such as CCK, **Display Suite** and **Views**. These modules proved to be extremely versatile at being able to accommodate the varying requirements of each entity. Multiple modules were used and thus the Effort level is 3. The ease of implementation is rated as moderate since the implementation was straight forward but did require the investigation of a number of modules. It should be noted that a significant amount of effort is needed to master the concepts related to CCK, but following this the implementation was not difficult.

- **Capability:** Yes
- **Effort level:** 3 (Installation of multiple add-on modules)
- **Ease of implementation:** Moderate

This section discussed the implementation of all domain entity requirements. Table 7.3 summarises the results of the classification according to Effort level and ease of implementation.

TABLE 7.3: Summarised results for domain entity requirements

No	Requirement	Effort level	Ease of implementation
D1	Course	3	Moderate
D1	Bursary		
D3	Facilities		
D5	News		
D6	Poll		
D7	Postgraduate Student		
D8	Publications		
D9	Research Groups		
D10	Staff		
D13	Vacation Work		

7.4 Implementation of domain non-entity requirements

The remainder of the requirements refer to concepts that fall within the domain layer but do not represent entities.

7.4.1 D4: Generic pages

This requires that pages be created that describe various topics. Examples given in the specification are a history page, an alumni page, pages for student activity and student groups. These pages should support text, images and video. The text should be capable of being formatted into headings and sections.

The page is a basic component of the WCMS and thus is a feature that is supported out of the box. However in order to support text formatting, images and video, a number of

additional modules were needed. In the case of image support a number of competing modules exist; thus effort was expended to determine which module was best suited and functioned as expected. Consequently the ease of implementation is classified as moderate.

- **Capability:** Yes
- **Effort level:** 3 (Installation of multiple add-on modules)
- **Ease of implementation:** Moderate

7.4.2 D11: User groups and access control

It was required that users of the system be assigned into groups and each group be assigned suitable access rights. The initial groups anticipated were academic staff, administrative managers and news editors. Academic staff were given rights to maintain their own profile pages as well as courses they co-ordinated. The administrative manager was given rights to create and maintain course information, general student information pages and publications. The news editor was assigned to manage the publications of news items.

User accounts, groups and access control are part of the Drupal core. The core ensures that security is pervasive through the application by ensuring that each module grants permissions per group. Thus when a new module is activated it needs to be configured for access control before it can be used. This measure ensures that no features are inadvertently active for the incorrect group of users. In order to make use of access control, configuration of the appropriate modules is necessary.

- **Capability:** Yes
- **Effort level:** 1 (Configuration of core)
- **Ease of implementation:** Easy

7.4.3 D11: User management

As the website will have a number of users, it is necessary for the application to have a workflow to support the creation and maintenance of user accounts. Drupal core

supports different modes of user account creation. Users can choose to register, and their requests will be forwarded to a site administrator for approval. Alternatively account self creation may be disabled and only created by a site administrator. The core also integrates to email services so that it can support traditional administrative tasks such as password reset via email.

User accounts are part of the Drupal core. As such the management of user accounts are straight forward and simple.

- **Capability:** Yes
- **Effort level:** 0 (Present by default)
- **Ease of implementation:** Easy

This section discussed the implementation of all domain non-entity requirements. Table 7.4 summarises the results of the classification according to Effort level and ease of implementation.

TABLE 7.4: Summarised results for domain non-entity requirements

No	Requirement	Effort level	Ease of implementation
D4	Generic Pages	3	Moderate
D11	User groups and access control	1	Easy
D12	User management	0	Easy

7.5 Implementation of infrastructure requirements

Infrastructure requirements refer to those that provide basic services to applications. These services are not domain specific but rather are common to a range of applications. Common examples include services such as persistence and logging. This section will discuss the infrastructure requirements for the school web application, elaborating on how the implementation was carried out.

7.5.1 I1: Backup

It is required that the application configuration and data be backed up, in order to facilitate a restore in the event of a disaster. Drupal stores configuration in both the file system as well as a database.

One approach to backup is to use operating system scripts to archive Drupal files and the database. However this may be problematic since the method of restoring the application is unclear. Simply copying the requisite files to a new host may be insufficient, and could lead to inconsistencies. For this reason, it is preferable to use a backup method that is part of Drupal. **Backup and Migrate** is an add-on module that is built for this purpose. This module permits a periodic task to trigger a backup operation and thus automates the backup process. The migrate tool permits data to be moved to a new site.

- **Capability:** Yes
- **Effort level:** 2 (Installation of single add-on module or theme)
- **Ease of implementation:** Easy

7.5.2 I2: Email

It is necessary for the application to generate emails. This may be for user account management as well as for system notifications to users.

The email functionality is part of Drupal core. Thus no additional module is required. It is however necessary to install operating system libraries and to configure an appropriate email server which supports Simple Mail Transfer Protocol (SMTP). Thus the implementation was straightforward and rated as easy.

- **Capability:** Yes
- **Effort level:** 0 (Present by default)
- **Ease of implementation:** Easy

7.5.3 I3: Image display

Images are frequently used across the site and thus a standardised mechanism for handling them is required. Besides supporting images within pages, it is also required to crop and resize images for pages which require a fixed size, such as the staff profile photographs.

This feature was not present in Drupal core and thus an add-on module was sought. A number of alternative modules were tested, but eventually `imce` module was chosen, as it best fit the requirements and was easiest to use. The configuration of this module was complex and in some cases it was difficult to resolve issues. As such the ease of use is rated as moderate.

- **Capability:** Yes
- **Effort level:** 2 (Installation of single add-on module or theme)
- **Ease of implementation:** Moderate

7.5.4 I4: Input sanitation

Web applications are inherently publically accessible and thus are vulnerable to attack. Frequently attackers attempt to upload malicious code into form inputs, and as a result disrupt the service of an application. Consequently it is necessary to protect against this type of attack by sanitising the input. This means that the input text is filtered to remove any unwanted parts. A typical filter would only permit a subset of HTML tags to be entered and eliminate all Javascript code.

Drupal core implements input filters that validate the data entered. A number of built-in filters exist that can be extended to provide for individual requirements. When a page or entity is created, it is necessary to specify the type of filter to be applied.

- **Capability:** Yes
- **Effort level:** 0 (Present by default)
- **Ease of implementation:** Easy

7.5.5 I5: Link Checker

A large website can consist of hundreds of pages, each containing many links. Internal and external links can break, and thus it is necessary to periodically check if all links are still valid. Broken links are very annoying to end users and thus must be prevented.

A link checker is not provided within the core application and thus it was necessary to find an add-on module. The **Link Checker** module provides this capability. It can be scheduled to run periodically and provide a report on broken links to the system administrator.

- **Capability:** Yes
- **Effort level:** 2 (Installation of single add-on module or theme)
- **Ease of implementation:** Easy

7.5.6 I6: Logging

Logging is an important part of most web applications since it permits the monitoring of the health of applications, as well as help debug error conditions.

As can be expected, logging is a key function and thus is implemented within the Drupal core. It has default settings but can be customised by the administrator.

- **Capability:** Yes
- **Effort level:** 0 (Present by default)
- **Ease of implementation:** Easy

7.5.7 I7: Popular links

It is required that the web application monitor access to specific pages and then dynamically produce a list of the most popular links. This list will be displayed within a block on the homepage.

Drupal core tracks all traffic to the application and thus is able to provide a list of popular links. In order to display the popular links on the homepage, it is necessary to configure a display block (as described in Section 7.1.4). The implementation is straight forward and easy to implement.

- **Capability:** Yes
- **Effort level:** 1 (Configuration of core)
- **Ease of implementation:** Easy

7.5.8 I8: Private pages

Private pages are those that are only visible to a selected group of users. The motivation behind this requirement was to provide a private document repository that would only be visible to staff members. Any other person attempting to access this page would receive an error message.

Private pages are not available within the core and thus an add-on module was sought. The module **Private** was found to be suitable and was installed. This module permitted a page to be marked as private and thus restricted its viewing from the public. The disadvantage of this module was that it is not possible to define separate categories of private pages, that would only be accessible to a particular group. Instead a page could only be flagged as private or not. Consequently access to the private page would then be granted to all groups that had access rights.

- **Capability:** Yes
- **Effort level:** 2 (Installation of single add-on module or theme)
- **Ease of implementation:** Easy

7.5.9 Regular tasks

It is required that a number of house-keeping tasks run regularly. This includes clean up of temporary files, re-indexing for search, and rebuilding of the cache. Consequently it is useful to have a standard scheduler in place.

Drupal supports periodic tasks natively within the core application. It provides a specific URL that when contacted, will perform all tasks. This is useful since it can be integrated with an operating system schedule task such as a `cron` job. It also supports a lazy mode, whereby a new interaction with the application will first trigger the periodic task to run if it has not done so recently. This is beneficial since certain hosting solutions do not provide access to the operating system.

- **Capability:** Yes
- **Effort level:** 0 (Present by default)
- **Ease of implementation:** Easy

7.5.10 I10: Rich text editors

The text displayed on a web page is authored using HTML markup. This is a complex format that is not suited to the lay person and thus a high level editor is needed to abstract away the details of the underlying HTML format. A rich text editor is a web browser component that permits users to edit content on the website by making use of a palette of commonly used functions. These functions include choosing fonts, marking text as headings, and including images into the document. Such an editor is highly desirable since authors no longer need to know any HTML. Rich text editors offer the editor a real-time view of what the final content will look like, and thus are also called What-You-See-Is-What-You-Get (WYSIWYG) editors.

This feature is not available within Drupal core and thus the add-on module was sought. Following testing of alternatives, `CKEditor` was eventually chosen as the preferred implementation. `CKEditor` is a general purpose rich text editor that is used in a number of web applications. The implementation was problematic as the module did not perform as expected. The debugging process was problematic as it was difficult to determine if the problem existed within the generic `CKEditor` component or within the Drupal module that contained it. When such a component malfunctions there is very little diagnostic information that is available, and thus fault finding can be difficult.

- **Capability:** Yes

- **Effort level:** 2 (Installation of single add-on module or theme)
- **Ease of implementation:** Difficult

7.5.11 I11: Search

For a large site, it is often difficult for users to find content purely through menu navigation. Consequently it is useful to implement a site wide search function. The search should be as comprehensive as possible and thus include all content types including domain entities defined.

Search is a built-in feature of Drupal and the search box is present by default in most themes. The CCK family of modules (as discussed in Section 7.3.12) integrates with Drupal search functionality, thus making all domain entities searchable. This is crucial since most of the site content is defined using domain entities.

For the search index to be up-to-date it is necessary to run a periodic task that indexes new items added to the site. Drupal supports lazy periodic tasks, whereby the task is run when a page request reaches the web server, or eager periodic tasks, whereby the task is executed on a fixed time schedule. The lazy mode is suited to site maintainers who do not want to go to the trouble of setting up a `cron` job, but instead want a simpler solution. For the school website it was decided to use the eager mode, since there already existed a periodic back-up task.

- **Capability:** Yes
- **Effort level:** 1 (Configuration of core)
- **Ease of implementation:** Easy

7.5.12 I12: Search engine optimisation

Search engines perform indexing and ranking by visiting web pages and inspecting the content, commonly referred to as crawling. Besides indexing the page based on the content visible to end-users, they also inspect the meta-data included in the page header [55]. In order to receive a good rank within a search engine, it is necessary to correctly author the appropriate meta-data.

The module **Page Title** was used as it permitted the modification of the appropriate page information such that it improved search engine indexing. It also permitted the generation of the appropriate data for domain entities, thus eliminating the need to optimise new entity pages that were added. This approach used tokens and pattern matching, so that a generic formula could be defined for each domain entity. The implementation of this feature was easy since the feature was well documented.

- **Capability:** Yes
- **Effort level:** 2 (Installation of single add-on module or theme)
- **Ease of implementation:** Easy

7.5.13 I13: Site map

A site map is a page that shows all pages within the site, as defined by their menu hierarchy. This is often used to locate pages that cannot be found through conventional means. Since the website grows organically based on the contributions of a number of users, it would difficult to perform this task manually, but rather it would be best to provide a dynamically generated map.

An add-on module provides the requisite site map functionality. The inclusion of this module was simple and did not pose any difficulties.

- **Capability:** Yes
- **Effort level:** 2 (Installation of single add-on module or theme)
- **Ease of implementation:** Easy

7.5.14 I14: Site statistics

Statistics on the access to individual pages or the site at large is very useful for content editors. Frequently the head of a research group would like to know how many hits a particular page is receiving, as well as referring web page which directed the user to the current page.

Drupal core provides for a range of statistics to be generated. The site administrator can customise both the type and detail of information that is available. In certain cases, extensive logging may slow the page response times and thus should be used with caution.

- **Capability:** Yes
- **Effort level:** 1 (Configuration of core)
- **Ease of implementation:** Easy

7.5.15 I15: Site structure export

Search engines infer the site structure by considering how pages link to each other within a site. However this information is not explicit and thus errors may occur. For this reason, web masters are encouraged to upload a site structure map to a search engine to help with this indexing. The Google search engine uses Webmaster Tools [56]. The Webmaster tools expect the site map in a specific XML format, and thus the application should generate such a file.

This feature is implemented using the module `XMLSiteExporter`. The module generates the appropriate XML file which can then be uploaded to the appropriate search engine. It also operates in a fully autonomous mode whereby it is able to directly submit the new site map to the search engine provider on a periodic basis.

- **Capability:** Yes
- **Effort level:** 2 (Installation of single add-on module or theme)
- **Ease of implementation:** Easy

This section discussed the implementation of all infrastructure requirements. Table 7.5 summarises the results of the classification according to effort level and ease of implementation.

This chapter elaborated on the details of how each requirement was implemented. Each requirement was classified into an effort level, which indicates how the implementation

TABLE 7.5: Summarised results for infrastructure requirements

No	Requirement	Effort level	Ease of implementation
I1	Backup	2	Easy
I2	Email	0	Easy
I3	Images	2	Moderate
I4	Input sanitation	0	Easy
I5	LinkChecker	2	Easy
I6	Logging	0	Easy
I7	Popular links	1	Easy
I8	Private pages	2	Easy
I9	Regular tasks	0	Easy
I10	Rich text editors	2	Difficult
I11	Search	1	Easy
I12	Search engine optimisation	2	Easy
I13	Site map	2	Easy
I14	Site statistics	1	Easy
I15	Site structure Export	2	Easy

was completed. In addition, specific challenges were discussed and used to as a basis for measuring the ease of implementation.

Chapter 8

Application testing, deployment and operation

This chapter discusses the application testing, deployment and operation. The testing was guided by the work of others in testing off-the-shelf software. Testing considers functional testing and non-functional testing (security and performance). The application deployment process is discussed, and the chapter closes with discussing failures in live operation.

8.1 Approaches to testing WCMS applications

A survey of the literature did not reveal any work specifically on testing WCMS applications. The case studies and books in the literature [32], [34], [35] do not provide any significant coverage on methods to testing WCMS. As a WCMS is an example of off-the-shelf software, the testing methods used in this area was considered. Morisio et al. conclude that most projects in their study spend testing effort on integration testing and black box testing [44]. Furthermore testing is done incrementally as each feature is added to the system. Black box testing refers to testing only input and output, as the internals are hidden to the tester [57].

8.2 Functional testing

The approach suggested by Morisio et al. above was adopted for the school website development. It was found that incremental development and testing worked well. It was natural to configure or install a module, and then test its functionality. Thus the requirements were implemented incrementally and each new implementation was accompanied by testing.

Testing mainly consisted of verifying the expected functionality for a given module. The verification would be done manually where a number of specific tasks were executed via the user interface. As an example, verification of the Course entity would consist of tasks such as:

- Create, read, update and delete (CRUD) a course
- Provide a listing of all courses
- Provide filtered listing based on a specific criterion (program of study)
- Test the integration of the course with other related entities e.g. Lecturer

8.3 Limited testing needed when compared to bespoke development

The testing effort for the WCMS differed significantly compared to bespoke development. Morisio et al. noted that teams assembling COTS software performed fewer verifications, as it was felt that the systems were being composed of pre-tested components [44].

In bespoke development testing has to be extensive as it is necessary to verify if the software fulfilled the requirement, and the implementation was free of error. As the WCMS was built using modules which have been tested extensively in other projects, it was not needed to thoroughly verify if the implementation was free of error.

An example may help clarify the above distinction. Consider the development of a calculator library, which performs the basic arithmetic functions (addition, subtraction,

multiplication, and division). If this was being done in a bespoke project, it would be necessary to consider if the calculator's functionality was sufficient for its intended use. Furthermore, it would be necessary to verify that the implementation of addition, subtraction, multiplication, and division were bug-free.

In contrast for the WCMS, it was mostly necessary to verify if the module was able to provide the expected behaviour. If this was the case, it was not necessary to extensively test if the implementation was correct, as it was assumed that a mature and widely used module functioned correctly.

In many instances, the verification of all fields was unnecessary. Consider as an example the field `Program of study`. This may be a fixed list consisting of options `Electrical Engineering` and `Information Engineering`. With a hand written application a significant amount of testing would be needed to check that the constraints for this field behave as expected. However with the CCK field, it is reasonable to assume that the implementation is error-free, as there is a generic mechanism for handling option lists. As such it was not necessary to verify fixed listing fields when they occurred in other entities. Rather it was necessary to verify that fixed listing fields worked in a single instance, rather than all instances.

The above discussion may give the impression that it was assumed that all module behaviour was bug free. This was not the case and functionality was verified for each new functionality within a module. The reduction in testing was due to the fact that the same module was used for implementing many requirements, and thus it was not necessary to verify the module behaviour each time.

It may be considered problematic that the application constructor cannot perform unit tests. Whilst it is true that the system is closed off to application constructor, it does not imply that unit testing is absent. Rather the task of unit testing falls to the module developers of the system, who are responsible to ensure that their modules behave as expected and are bug free. In the case of open source modules, the unit test results are available online.

Following the above-mentioned tests, a pre-release version was made available to members of staff within the school. The staff provided feedback on usability and also were able to pick up bugs that were not initially detected.

8.4 Non functional testing

8.4.1 Page load times

The page load times of the ten most popular pages were measured from a client on the same local area network as the web server. This was done to eliminate any delays due to transport over the Internet. The average page response time is 378.3 milliseconds with a standard deviation of 77.6 milliseconds. The measured values are shown in Table 8.1 Galetta et al. reported that the commonly held web page delay ‘gold standard’ is 2 seconds or less [58]. User experience researcher Jakob Nielsen stated that users do not feel that they are waiting on the computer if the response time is below one second [59]. The application’s response times are within these guidelines, though it must be noted that Internet transport delay is excluded.

TABLE 8.1: Response times for the ten most popular pages on the application

URL (relative to www.eie.wits.ac.za)	Page load time (milliseconds)
/	417
/undergrad/current-student-info	429
/staff	485
/research/biomedical	385
/postgrad/current-student-info	368
/research/telecoms	259
/research/software	404
/research/energy	393
/research/control	214
/research/high-voltage	429
Average	378.3
Standard deviation	77.6

During the implementation it was noted that pages loaded slowly if large images were used. It was thus decided to follow web guidelines to optimize images [60]. This involved using an appropriate format (`jpg`, `png`) dependent on content type, and ensuring file sizes and resolutions were within guidelines.

8.4.2 Website load

The application has been extensively used during the period of live operation, and receives approximately 4500 hits per month. Although this traffic load is modest compared

to many web applications, this result does indicate that Drupal is able to adequately handle reasonable traffic volumes. The application was configured to trigger alarms when load is outside of tolerances. During the operational period there have been no alarms triggered to indicate load problems. Drupal's ability to handle high traffic volumes is not considered in this study, as the application under development is considered a low traffic site.

8.4.3 Security

Meike et al. have studied the web security practices in WCMS [40] by examining the security of Drupal and Joomla in 2009. Their results are summarised below.

- WCMS are targets for attacks owing to their ubiquity
- Both platforms have dedicated teams who work on security aspects
- The large community helps to spread news of vulnerabilities quickly, resulting in rapid fixes
- The platforms do provide mechanisms to alert administrators of pending security updates
- Vulnerabilities are easily introduced through the addition of modules
- An inexperienced or non technical user can create security problems through poor configuration

Patel et al. performed a similar study on Drupal, Joomla and WordPress in 2013 [61]. The study subjected the WCMS to well-known attack vectors such as SQL injection, cross site scripting and directory traversal. Their findings were similar to Meike et al. and they remarked that the core product provided good basic security, but vulnerabilities existed within third party modules. They also noted that incorrect permissions on directories could lead to vulnerabilities.

In view of the above, the following security practices were implemented:

- Utilised Drupal security practice checklist [62]

- Utilised the **Security Review** module to check for vulnerabilities
- Periodically ran a web security scanner [63] to test for vulnerabilities
- Installed security updates to platform and modules

During the period of evaluation no known security breaches were detected.

8.5 Application deployment

After implementation, the application was then deployed to a live environment. The Drupal documentation was extensive and described the procedure for deployment [64]. The greatest challenge during this task was the configuration of the application stack. Drupal was deployed on a Linux machine which runs Apache web server, MYSQL database and PHP server execution environment, commonly referred to as the LAMP stack. This aspect seems to have been simplified and today many hosting providers support pre-configured environments that support easy installations (commonly referred to as one-click installers [65], [66]).

8.6 Failures during operation

The application has been observed for a 24 month period. During this period a small number of faults were logged. Table 8.2 summarises the errors encountered during this period. A total of 24 faults were logged during this period, 13 of which were related to the Drupal application. Each item will be expanded in the following sections.

8.6.1 Power failure

The majority of problems encountered were due to power failures in the server hosting environment. This led to service outages but was not due to the Drupal application.

8.6.2 Corrupt user session table

The application did experience a fatal error which was a result of a power outage. Upon reboot the platform was not able to start successfully. The error messages indicated

TABLE 8.2: Errors encountered during live operation over 24 month period

Item	Problem	Description	Severity	Root cause	Frequency
1	Power failure	Server unavailable due to power outage	Major	Server hosting environment	8
2	Corrupt user session table	Users are unable to login following a server startup.	Major	Drupal	4
3	Disk space exhausted	The system does not behave correctly when database log files fill disk	Major	MYSQL Database	3
4	Presentation rendering errors	A number of pages presented with layout errors due to theme selected	Minor	Drupal	1
5	Implementation errors	Incorrect content and menu errors	Minor	Drupal	8

a problem with user sessions that did not terminate successfully. Consequently these users were unable to login and create new data. This problem was eventually resolved after consulting user forums. The solution requiring a manual removal of data from a database table through the execution of a Structured Query Language (SQL) script [67]. This problem recurred whenever the system shut down unexpectedly, and became part of the standard restore procedure.

8.6.3 Disk space exhausted

A further problem encountered was the exhaustion of disk space, due to large log files created by the MYSQL database. This was resolved through configuration of the database and was not due to Drupal. It may have been a result of incorrect configuration by the system administrator.

8.6.4 Presentation rendering errors

It had been noticed that the theme used did exhibit errors in the layout of certain pages, particularly when it displayed Content Creation Kit (CCK) entities. This problem was not resolved and was most likely due to theme incompatibilities with specific modules.

8.6.5 Implementation errors

During the operational period implementation errors such as incorrect content and menu errors were detected. These errors were either reported by end users or detected by the site administrator during content updates.

8.7 General maintenance

Maintenance work on the application revealed a deficiency in the documentation of how features were implemented. In a traditional coded application, developers make use of code comments to describe the specifics of the implementation, and as a guide for future work. In contrast the only artefact from installing and configuring modules is the final product. Thus it was necessary to create separate documentation to guide maintenance.

This chapter focused on the testing, deployment, and live operation of the school WCMS application. Testing practices were guided by the literature and consisted of black box testing. The application was found to perform within recommended limits for page response times, and did not exhibit any problems due to traffic load. Security audits and updates were performed regularly; and no known breaches were detected. The chapter closes with consideration of failures that occurred during live operation within a two year period.

Chapter 9

Results and Discussion

This chapter summarises the results of applying the methodology to the developed application. The analysis is performed for each of suitability indicators: capability, effort and ease of implementation. For each indicator overall trends are considered, as well as trends per requirement group (presentation, domain and infrastructure). The findings do point to some possible trends, but this can only be tested with further use of the methodology on other applications. The findings do validate that the methodology can be applied successfully.

9.1 Evaluation of capability

Capability is the indicator of whether the WCMS platform is capable of realising a requirement. For each requirement in the application, it was possible to assign a classification as given below:

- Yes (Requirement can be implemented)
- No (Requirement cannot be implemented)

It was found that 100% of requirements were fulfilled by the WCMS platform (see Figure 9.1). This is a positive indication of the WCMS capability, but it must be noted that this is only a single result for an individual application. Application of the methodology to a wide range of applications would provide a better indication of the platform capabilities.

This result does indicate that the methodology can be successfully applied to measure capability.

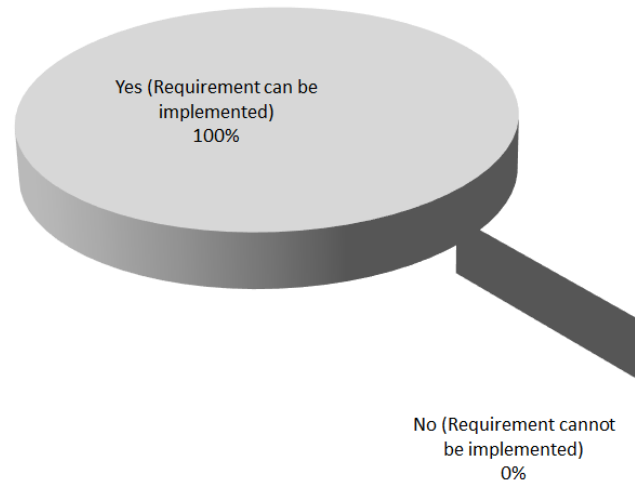


FIGURE 9.1: Percentage of requirements that the WCMS platform was capable of realising

9.2 Evaluation of effort level

The implementation of each requirement was classified into an effort level (as described in Section 5.4).

9.2.1 Analysis of overall effort level results

The initial analysis considered the percentage of requirements that were implemented per effort level, as this provides a high level decomposition of how the requirements were implemented. As noted in Figure 9.2, 16% of requirements were present by default in the core, 22% required some configuration of the core, 32% required a single add-on module to be installed, and 30% required multiple add-on modules to be installed. None of the requirements necessitated the coding of a theme or the coding of a module.

It is interesting to note that the entire application was implemented without the need for writing any custom code. Thus 100% of requirements were implemented through

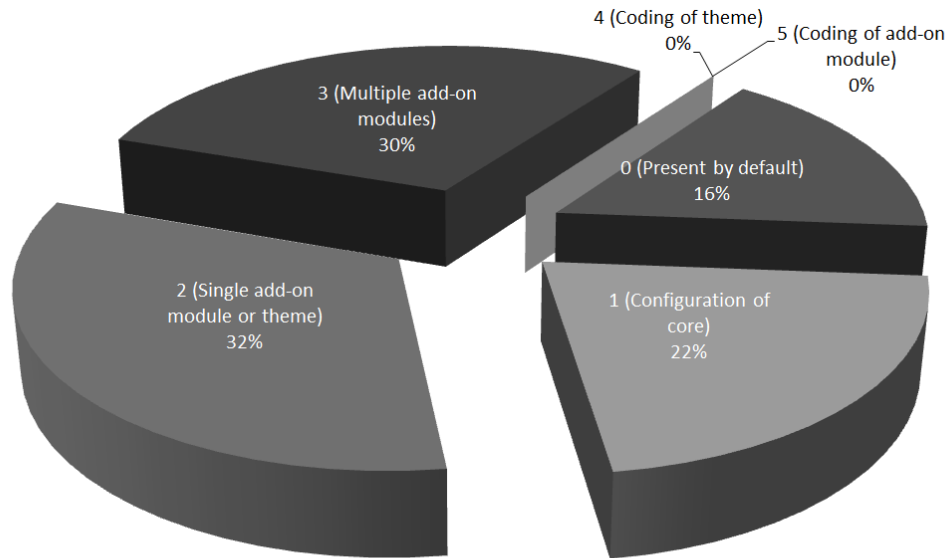


FIGURE 9.2: Percentage of requirements implemented per effort level

configuration of either core or add-on modules. This result is somewhat surprising since the requirement list is substantial and many requirements were specific to the chosen domain. In this instance the WCMS ecosystem provided sufficient variety. It would be useful to see how the WCMS copes across a range of applications.

9.2.2 Split of core and add-on modules used for implementation

Level 0 and 1 refers to requirements that were implemented using features within the core application, whilst Level 2 upwards refer to implementations employing add-on modules. If we consider this split, then 38% of requirements were fulfilled using core features whilst 62% was implemented using add-on features (see Figure 9.3).

9.2.3 Analysis of working with many third party modules

Given that the majority of the features were implemented using third party modules, it is reasonable to expect that there might be integration issues and latent bugs. In certain instances it was difficult to integrate modules, but for the most part integration and operation was straightforward.

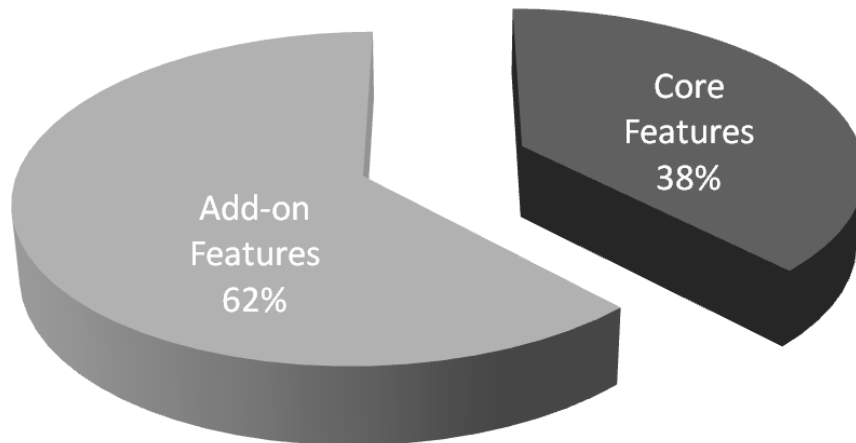


FIGURE 9.3: Split of core and add-on modules used for implementation

The process of finding modules to suit a task did pose challenges in certain instances. For simple and explicit features such as `Menu breadcrumbs` (see 7.1.8), it was usually easy to find a module by searching generally on the Internet or Drupal module listings. For features that usually involved the combination of a number of modules, finding the appropriate module/s was more challenging. In these cases it usually involved experimentation and reading of tutorials to resolve the issue. The CCK modules in particular required extensive learning in order to understand the capabilities and constraints of use.

9.2.4 Effort levels for presentation requirements

Presentation requirements were defined as those that impacted on the graphical layout of the application, the look and feel of the application, and the site structure (menus, URLs). For this category 12% of the requirements were present by default within the core, 44% required configuration of the core, and 44% required a single add-on module. None of the requirements required multiple modules for their implementation (Level 3), indicating that modules satisfying presentation requirements tend to be stand-alone in this instance.

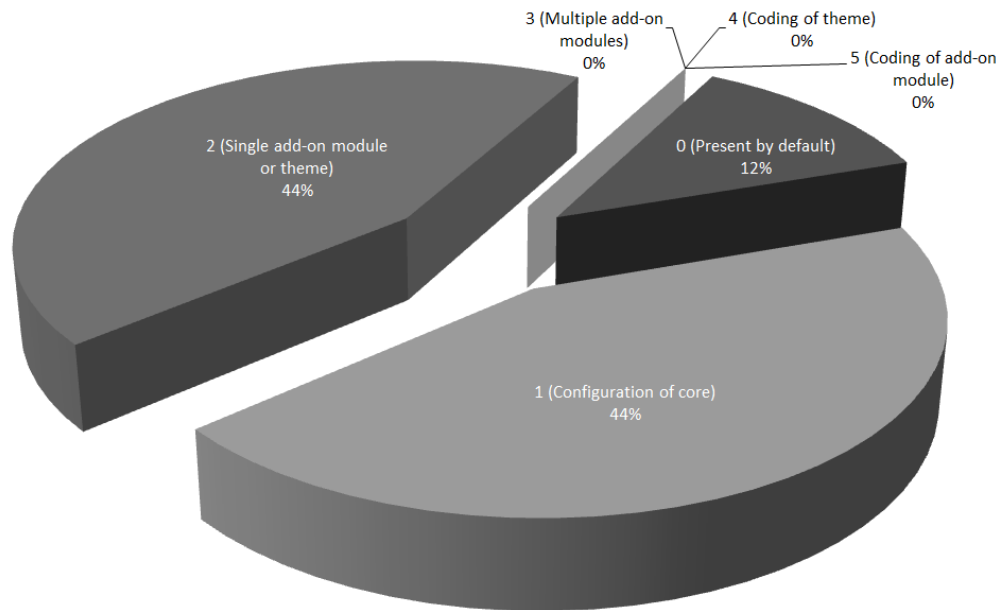


FIGURE 9.4: Effort level split for presentation requirements

When considering the presentation requirements defined, it is apparent that some requirements represent more fundamental functionality, whilst others are secondary in nature. Requirements P1 (Clean URLs), P4 (Footer), P5 (General page layout), P6 (Menu) and P9 (Theme) would be considered fundamental since they address the basic building blocks of the presentation of a website. P2 (Dynamic Menus), P3 (External Links), P6 (Homepage layout) and P8 (Menu breadcrumb) are secondary and represent the specific design choices for a site.

An analysis of the effort results, as shown in Table 7.1), indicates that the fundamental requirements are Level 0 and Level 1, and thus part of the core application. This is expected since it indicates that fundamental aspects have been successfully included in the core application whilst additional features are developed as add-on components.

A key enabler of flexibility in presentation is the `block` system, as described in 7.1.4. It allows regions to be defined that can be easily assigned to a specific location within the page. This provides a degree of decoupling and allows pages to be constructed via a configuration process.

9.2.5 Effort levels for domain requirements

Domain requirements refer to those that are specific the chosen application domain, which in this case is the school website. The analysis of domain requirements, as discussed in Section 6.3.2, partitions the requirements into domain entities and domain non-entities. Domain entities refer to collections of objects that are found in the domain. Non-entities are all other remaining requirements. Each of the these subgroups will be analysed separately.

9.2.6 Effort levels for domain entities

The implementation of all domain entities followed the same pattern - the Content Creation Kit (CCK) family of modules were used. Each requirement necessitated the use of a combination of modules (CCK fields, Views, Display Suite) and thus the effort levels for this subgroup of requirements is Level 3 - Multiple add-on modules used. As shown in Figure 9.5 100% of requirements were fulfilled using multiple modules.

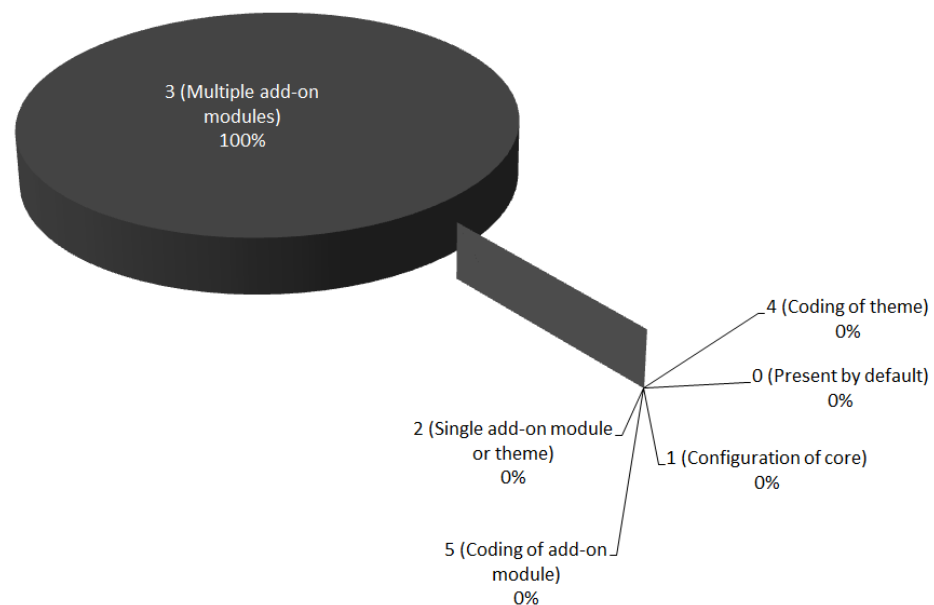


FIGURE 9.5: Effort levels for domain entity requirements

9.2.7 Effort levels for domain non entities

In this subgroup, there were only three requirements. Consequently it is difficult to draw conclusions from such a small sample.

9.2.8 Effort levels for infrastructure requirements

The infrastructure group of requirements are those that are general purpose in nature, and thus applicable to range of applications. Figure 9.6 shows the distribution for the infrastructure requirements. Category 0, 1 and 2 accounted for 27%, 20% and 53% of the requirements respectively. The split between core and add-on features is 47% and 53% respectively.

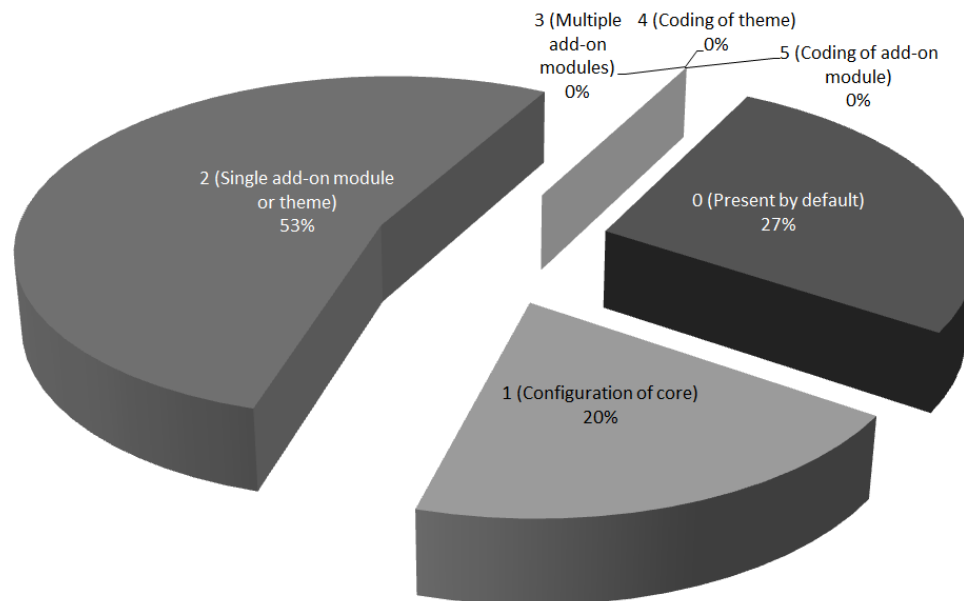


FIGURE 9.6: Effort levels for infrastructure requirements

With infrastructural features, it is expected that a significant portion should exist within the core of the application, since this is functionality that could be used by a number of different applications. The finding that 47% is within the core is reasonable, and it is anticipated that future versions will incorporate further features into the core. Indeed, Drupal 7 has taken further steps to increase the core functionality [68, p. xlvii].

9.3 Evaluation of ease of implementation

In addition to effort, each requirement was also rated on the ease of implementation. Whilst a requirement may fall into a specific effort level, it is also necessary to consider particular difficulties with the implementation. This metric would help to identify deficiencies within particular modules and areas that need improvement.

9.3.1 Overall ease of implementation

For the application as a whole, the ease of implementation distribution is given in Figure 9.7. 54% of requirements were considered as easy to implement, 41% as moderate and 5% as difficult. Following on from this analysis, strengths and weaknesses of the WCMS have been identified. It is stressed that the strength/weakness analysis is based on a single implementation, and thus are not conclusive.

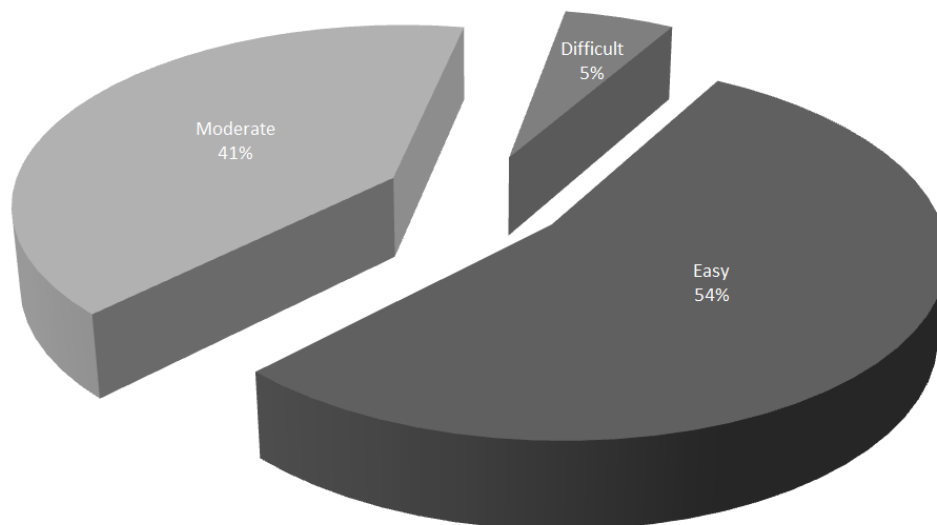


FIGURE 9.7: Ease of implementation for overall application

Based on the current implementation, the WCMS strengths are:

- **Maturity** - Drupal has gone through a number of versions, with each version adding to the maturity of the product [3, p.427]. This has meant that the product has become more stable. A stable and mature platform means that fewer bugs are encountered and modules work well together.

- **Adoption** - The platform is widely adopted and continues to grow [27]. This means that the number of contributors and users is growing and thus keeps the product viable and vibrant.
- **Documentation** - There is a large variety of documentation (books, tutorials, videos) for Drupal [3, p.15]. In addition, users contribute to forums, which are a valuable resource to all involved.
- **Standardised administrative interface** - Drupal has standardised the pages that are used to administrate modules (see Section 2.9.3). As such system administrators become familiar with the application structure and can easily manage newly added modules. This familiar structure also ensures that module developers implement all the requisite functionality demanded by the platform.
- **Flexibility** The CCK family of modules were found to be flexible. They were used to implement all the domain entity requirements.
- **Modular approach** The overall ease of implementation showed that the modular approach adopted is sound.
- **Good integration** It was surprising to note how well add-on modules integrated into the core. The majority of modules installed and worked without fault.

The primary weaknesses identified are:

- **Choosing the correct module** - In many cases a number of alternative implementations of a particular feature exist. In this case it can be difficult to choose the appropriate module to use. In making this choice the user can consider factors such as the number of sites using the module, the rating of the module, and the date of the last maintenance release. In many cases, the documentation of modules is poor and thus it is difficult to determine if it is appropriate to the user's needs. The poor documentation also makes it difficult to troubleshoot a problem.
- **Learning curve** - Drupal is often regarded as a more challenging platform compared to its primary competitors [69]. Many regard it as a platform suited to software developers and its architecture is geared towards developer extension of core functionality. With this in mind, many novice users struggle to master the

fundamentals, especially when it comes to the sophisticated and complex CCK family of modules.

- **Debugging workflow** - It frequently occurs that a newly installed module that does not seem to work as expected. In many instances, there are no error messages returned and thus diagnosing the fault can be tedious. This often involves extensive searches on online forums. As such the process for debugging errors with add-on modules needs to be improved.
- **Poor Documentation** - Many modules used have poor documentation. As such it is difficult to resolve faults that occur.
- **Unmaintained and abandoned modules** - A large number of contributed modules often mean that a number of them often go unmaintained or are abandoned. This can be problematic if a particular project is highly dependent on such a module. The open source nature however is some salvo in that the further maintenance can be continued by a different party if so desired.

9.3.2 Ease of implementation for presentation requirements

For the presentation requirements, 55% of requirements were classified as easy, 33% as moderate and 11% as difficult (see Figure 9.8). The highest percentage of difficult requirements were found in this group, and can be attributed to issues integrating modules. During the implementation of the dynamic menu, the page layout and the homepage scrolling banners, various incompatibilities were detected. Frequently the theme was found to be incompatible with the dynamic menu module or with the dynamic display block. This was eventually resolved through trial and error, but is a non-ideal solution.

Whilst most aspects of the WCMS do not require any expert knowledge, it seems that the presentation layer falls short in this regard. The customisation options provided by the WCMS is extensive but fine grained control requires the user to know Hyper Text Markup Language (HTML) and Cascading Style Sheets (CSS). However it must be acknowledged that the majority of requirements were rated as easy and thus there are many presentation modules that work without any issue.

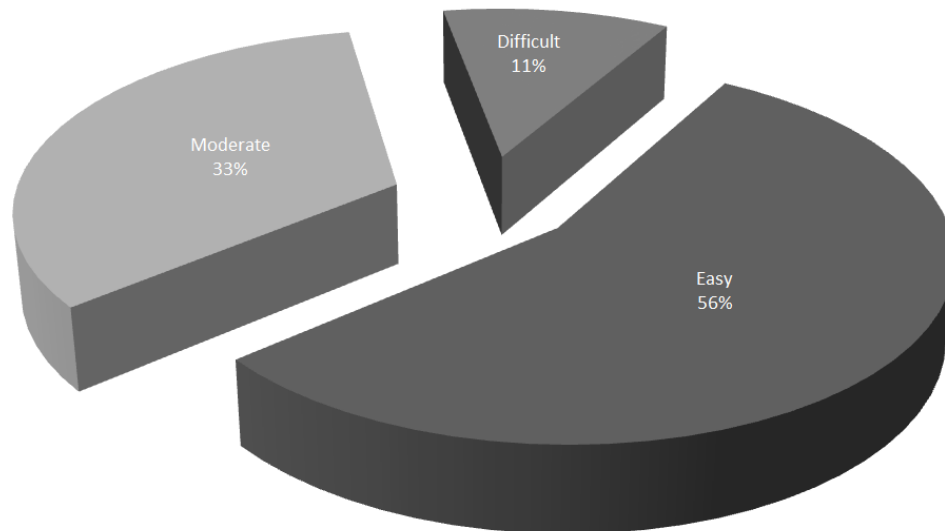


FIGURE 9.8: Ease of implementation for presentation requirements

9.3.3 Ease of implementation for domain requirements

Figure 9.9 shows the ease of implementation distribution for domain requirements. 85% of the requirements were classified as moderately difficult, and 15% were easy. This rating is based on the steep learning curve in using the CCK family of modules. These modules are complex and sophisticated in their design as they permit a variety of uses. The trade-off for the sophistication is that the novice user may struggle to grasp the concepts initially. As an example of this complexity, the **Views** module uses a concept of inherited behaviour to define views that have similar data but differ in minor details. This provides a significant reduction in configuration effort, but can often confuse a novice user.

The domain group encapsulates the aspects of the application that are unique to the domain and as such are often the most difficult to implement. It is interesting that no difficult requirements were found, and thus serves as an indicator of the value of the CCK approach. The reader is cautioned that this result is based only on one implementation, and thus the finding is anecdotal.

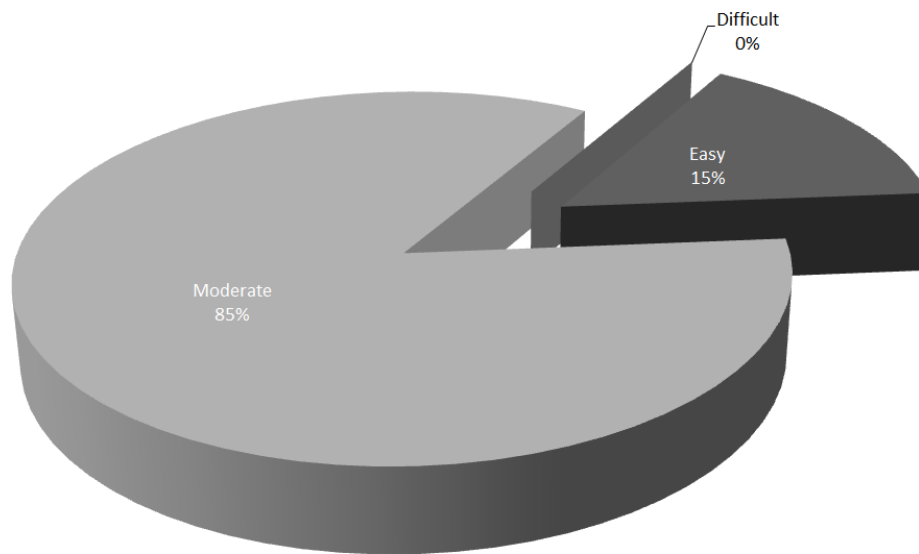


FIGURE 9.9: Ease of implementation for domain requirements

9.3.4 Ease of implementation for infrastructure requirements

The distribution for ease of implementation of the infrastructure requirements is 86% easy, 7% moderate and 7% difficult (see Figure 9.10). This result is positive and shows that the vast majority of infrastructure requirements were easy to implement. The only difficulty was the implementation of the rich text editor. The primary difficulty was that it was difficult to debug why the module did not behave as expected. The diagnostic information returned was poor and thus delayed the resolution of the problem.

9.4 Limitations of study

The aim of the study was develop and test a methodology for evaluating capability, effort and easy of implementation in a WCMS. This was achieved by constructing a custom web application and then drawing findings from the results of the project. The study was able to achieve these goals.

The reader is however cautioned that the results provided above are limited to a single application, and thus should conclusions cannot be broadly applied. It was never intended to draw broad conclusions from the analysis but rather to show that the methodology

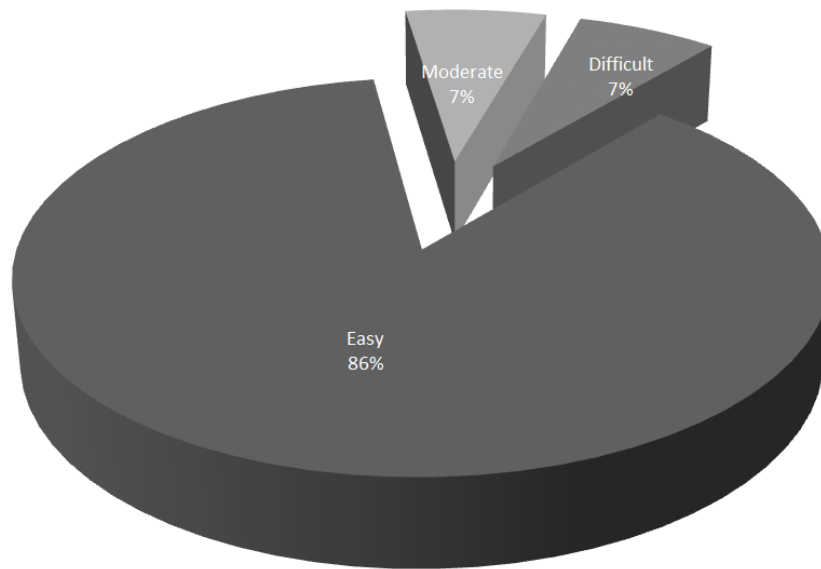


FIGURE 9.10: Ease of implementation for infrastructure requirements

was sound. Future work should consider extending the study to other applications, preferably in other application domains. Once a sufficient body of data is available, it would be possible to draw broader conclusions. It would also have been useful to compare the results to similar research done by others. Unfortunately no such research was found in the literature.

This chapter presented the results of applying the developed methodology to the school web application. The methodology prescribes the evaluation of the suitability indicators (capability, effort and ease of implementation). The results are analysed in terms of overall trends as well as trends per requirement group. The analysis of the results indicate that the methodology can be successfully used to evaluate an application. The chapter closes with a discussion of the limitations of the study.

Chapter 10

Conclusion

This study sought to develop and test a methodology to evaluate the suitability of using a WCMS as a platform for a given application. In this context, suitability was limited to three indicators: capability, effort and ease of implementation. The value of the methodology is that it would permit new implementers the ability to reference the experience of others who have implemented similar applications. With sufficient application of the methodology, a body of knowledge would be created that would provide general guidance as to where and when a WCMS should be used as an application platform. Furthermore the body of knowledge would permit an assessment of the strengths and weaknesses of the WCMS approach. In order to develop the methodology the study posed the following research question:

Can a methodology be developed and tested that would evaluate:

1. capability to realise a given application using a WCMS?
2. effort involved in implementing the application?
3. the ease of implementing the application?

The methodology was developed by considering work in the literature that evaluates capability, effort and ease of implementation. The methodology was tested by developing a school web application using Drupal WCMS, and then evaluating the suitability indicators for each of the application requirements.

It is stressed that the trends identified are based on this single application, and cannot be extrapolated to other cases. The study did not intend to produce trends that are widely applicable. The trend analysis is presented to validate that the methodology can be successfully applied for a given application. If the methodology is applied to a large sample of applications, it would be possible to identify trends that are widely applicable.

The capability indicator evaluates whether requirements can be realised. This indicator posed no difficulty to model as it was a simple binary evaluation; either the requirement could be implemented or not. For the given application, 100% of the requirements could be realised.

Defining an effort indicator was more challenging. Absolute measures of effort such as time are difficult to compare objectively. The time taken to complete a task can vary considerably depending on the implementer's skill, and can also vary due to external factors. Consequently a relative measure called the effort level was defined.

The definition of the effort level originated through the observation of the different modalities of building a WCMS application. Some functionality existed out of the box (within the core product) and required no or minimal configuration. Other functionality was only possible through the installation of one or more add-on modules and usually required more configuration. Simple features required configuration of the core or installation of a single add-on module, whilst complex features required the inter-working between a number of add-on modules and frequently required extensive configuration.

Thus effort was related to the number of modules used. Lastly if a particular functionality did not exist, it would require the development of a custom module or theme, which would require the most effort. Taking the above into consideration, a model of relative effort levels was defined as shown below.

- Effort level 0: Present by default
- Effort level 1: Configuration of Core
- Effort level 2: Installation of single add-on module or theme
- Effort level 3: Installation of multiple add-on modules
- Effort level 4: Coding of theme

- Effort level 5: Coding of add-on module

For the given application, the effort indicator evaluation showed that 16% of requirements were present by default in the core (Level 0), 22% required some configuration of the core (Level 1), 32% required a single add-on module to be installed (Level 2), and 30% required multiple add-on modules to be installed (Level 3). None of the requirements necessitated the coding of a theme (Level 4) or the coding of a module (Level 5).

The ease of implementation indicator was defined to assess particular difficulties that were encountered during the implementation. The classification is straightforward; each requirement is assessed as being easy, moderate or hard. In addition to the classification, commentary would be required to justify the decision.

For the given application, the ease of implementation evaluation showed that 54% of requirements were considered as easy to implement, 41% as moderate and 5% as difficult. Following on from this, strengths and weaknesses of Drupal within this context were identified.

A detailed analysis of trends identified is given in Chapter 9. The analysis also considered how the indicators vary per requirement group (presentation, domain and infrastructure).

The analysis was done to demonstrate that the methodology developed could be successfully used. It was not intended to draw broad conclusions from this single dataset. It would only be possible to examine general trends once the methodology is applied to more applications. It is thus recommended that future work apply the methodology to other applications in a variety of domains.

The study is concluded by noting that it was successful in achieving its aims of developing and testing a methodology to evaluate the suitability indicators: capability, effort and ease of implementation.

Appendix A

Detailed Application Requirements

This chapter provides a reference for the application requirements gathered from various stakeholders. The sections describe:

- Definition of properties for domain entities
- List of options for specific properties.

A.1 Definition of properties for domain entities

The tables listed below define the properties for each of the entities used in the domain requirements group. Each property is assigned to a primitive data type such as integer or text. In addition constraints on specific properties are listed. As an example an email field is a text property, but is further constrained to a specific format e.g. it must contain the @ symbol and is not permitted white spaces or other illegal characters.

In the case where a property can only be specified from a closed set of values, it is also necessary to specify what the permissible values are. This is described in [Section A.2](#).

TABLE A.1: Properties of Bursary

Bursary		
Property	Data Type	Constraint
Contact email	Text	Email validation
Is undergraduate	Boolean	None
Website	Text	URL
Telephone	Text	Telephone format
Attachments	File	None

TABLE A.2: Properties of Course

Course		
Property	Data Type	Constraint
Course code	Text	None
Course name	Text	None
School responsible for course	Text	Select from list
Course co-ordinator	Link	Staff member entity
Undergraduate course	Boolean	None
Year of study	Integer	None
Course brief and outline	Document	PDF format
Course image	Image	JPG, PNG format
Programs offering course	Checkbox	None
Syllabus	Text	Paragraph

TABLE A.3: Properties of News

News		
Property	Data Type	Constraint
Date published	Date	None
News image	Image	JPG, PNG format
Description	Text	Paragraph

TABLE A.4: Properties of Page

Page		
Property	Data Type	Constraint
Date published	Date	None
Information	Image & Text	Free arrangement of text and images
Author	Text	None

A.2 List of options for specific properties

Certain properties are constrained with the values that are permitted. As an example, some of the allowed titles for a person are Doctor and Professor. The tables shown below summarise the list of options for specific properties used in the application.

TABLE A.5: Properties of postgraduate

Postgraduate		
Property	Data Type	Constraint
Name	Text	None
Title	Text	Select from list
Photo	Image	None
Position	Text	Select from list
Research Group	Link	None
Interests	Text	Paragraph
Office	Text	None
Email	Text	Email format
Telephone	Text	Telephone format
Bio	Text	Paragraph

TABLE A.6: Properties of research group

Research Group		
Property	Data Type	Constraint
Research Description	Text	Paragraph
Research Head	Link	Staff member
Facilities	Link/s	Link to facilities

TABLE A.7: Properties of staff

Staff		
Property	Data Type	Constraint
Name	Text	None
Title	Text	Select from list
Photo	Image	None
Position	Text	Select from list
Research Group	Link	None
Interests	Text	Paragraph
Office	Text	None
Email	Text	Email format
Telephone	Text	Telephone format
Bio	Text	Paragraph

TABLE A.8: Properties of Vacation Work

Vacation Work		
Property	Data Type	Constraint
Contact email	Text	Email validation
Website	Text	URL
Telephone	Text	Telephone format
Attachments	File	None

TABLE A.9: Options for Schools responsible for course

Schools responsible for course
Accountancy
Anatomical Science
Animal, Plant & Environmental Sciences
Architecture & Planning
Biological and Life Sciences
Centre for Health Science Education
Centre for Postgraduate Studies and Research Office .
Chemical and Metallurgical Engineering
Chemistry
Civil & Environmental Engineering
Clinical Medicine
Computational & Applied Mathematics
Computer Science
Construction Economics & Management
Earth Sciences
Economic & Business Sciences
Electrical & Information Engineering
Geography, Archaeology & Environmental Studies
Geosciences
Graduate School of Business Administration (Wits Business School)
Graduate School of Public & Development Management
Human & Community Development
Humanities Graduate Centre
Law
Literature & Language Studies
Mathematical Sciences
Mathematics
Mechanical, Industrial & Aeronautical Engineering
Mining Engineering
Molecular & Cell Biology
Oral Health Sciences
Pathology
Physical Sciences
Physics
Physiology
Public Health
Social Sciences
Statistics & Actuarial Science
Therapeutic Sciences
Wits School of Arts
Wits School of Education
WitsPlus (BA for the World of Work)
WitsPlus (BCom)

TABLE A.10: List of options for staff positions

Position
Personal Professor
Professor
Professor Emeritus
Adjunct Professor
Associate Professor
Visiting Associate Professor
Senior Lecturer
Lecturer
Associate Lecturer
Honorary Lecturer
Research Fellow
Teaching Assistant
Admin Manager
Postgraduate Officer
Senior Administrative Secretary
Bookkeeper
Assistant Bookkeeper
Receptionist
Network Engineer
Power Workshop Manager
Electronics Workshop Manager
Senior Technician
Technician
Technical Assistant
Convergence Laboratory Manager
Retired

TABLE A.11: List of options for programs offering course

Programs offering course
BSc Eng (Mining)
BSc Eng (Mechanical)
BSc Eng (Metallurgy)
MEng (Telecoms)
MSc Eng (Telecoms)
MEng (Systems and Control)
MSc Eng (Systems and Control)
MEng (Information)
MSc Eng (Information)
MEng (Power)
MSc Eng (Power)

Appendix B

Implementation Details

This chapter provides a reference for the specific details of the application implementation. It will provide information regarding the modules that were tested and finally chosen for the implementation.

B.1 Modules used for implementing requirements

For the requirements that are classified as Level 1 to 3 it was necessary to use a third party module to fulfill the implementation (see Chapter 5). Table B.1 shows the modules along with their version numbers that were used. Full documentation on each of the modules can be found on <http://www.drupal.org>.

B.2 Listing of fields for domain entity types

For the implementation of each of the domain entities (as discussed in Section 7.2, it is necessary to define the fields that each entity contains. For each field it is possible to specify the data type (boolean, integer, list, document), constraints on the data type, and whether the field is mandatory or optional. The following tables list these fields and their associated characteristics.

TABLE B.1: Modules used for implementing requirements in Category 1-3

Module	Requirement	Module chosen
P2	Dynamic menu	jquerymenu-6.x-3.3
P3	External links	extlink-6.x-1.11.tar
P6	Homepage layout	ddblock-6.x-1.0-rc6.tar
P8	Menu breadcrumb	menu_breadcrumb-6.x-1.3.tar
P9	Theme	Theme: Pixture Reloaded
I1	Backup	backup_migrate-6.x-2.4.tar
I2	Email	email-6.x-1.2.tar
I3	Images	imce-6.x-2.0-rc2.tar
I5	LinkChecker	linkchecker-6.x-2.4.tar
I8	Private pages	private-6.x-1.1.tar
I10	Rich text editors	FCKEditor_2.6.6
I12	Search engine optimisation	page_title-6.x-2.3.tar
I13	Site map	site_map-6.x-2.2.tar
I15	Site structure Export	xmlsitemap-6.x-2.0-beta1

TABLE B.2: All fields for Course entity

Field Name	Data Type	Input format or Constraint	Mandatory (M) or Optional (O)
Course Code	Simple Text	Fixed format of 4 letters and four digits	M
Course Title	Simple Text	None	M
School Responsible for Course	List	Single choice	M
Course-coordinator	List	Single choice	M
Is Undergraduate Course	Boolean	None	M
Year of Study	Integer	1-5	M
Course Brief and Outline	Document	Allowed formats	M
Programs offering Course	List	Multiple choice	M
Display Image	Image	Allowed formats	M
Syllabus	Multi-line Text	None	M
Course website	URL	None	O

TABLE B.3: All fields for Bursary entity

Field Name	Data Type	Input format or Constraint	Mandatory or Optional
Name of Bursary	Simple Text	None	M
Description of Bursary	Multi-line Text	None	M
Contact Telephone Number	Simple Text	None	O
Contact Email	Email	Email validation	O
Website	URL	URL validation	O
Application form	Document	Attachment validation	O

TABLE B.4: All fields for Facilities entity

Field Name	Data Type	Input format or Constraint	Mandatory (M) or Optional (O)
Name	Simple Text		M
Description	Multi-line Text	None	M
Location Map	Image	Standard Formats	M
Location description	Multi-line Text	None	M
Opening Hours	Multi-line Text	None	O
Amenities	Multi-line Text	None	O

TABLE B.5: All fields for News entity

Field Name	Data Type	Input format or Constraint	Mandatory (M) or Optional (O)
Title	Simple Text		M
Description	Any Content	None	M
Teaser	Multi-line Text	None	M
Headline image	Image	Standard formats	M

TABLE B.6: All fields for Postgraduate Student entity

Field Name	Data Type	Input format or Constraint	Mandatory (M) or Optional (O)
Surname	Simple Text		M
First Name	Simple Text	None	M
Interests	Multi-line Text	None	M
Email	Email	Email validation	M
Research Group	List	None	M

TABLE B.7: All fields for Publication entity

Field Name	Data Type	Input format or Constraint	Mandatory (M) or Optional (O)
Name of Publication	Simple Text		M
Pages	Simple Text	None	M
Place	Simple Text	None	O
Type	List	None	M
Research Group	List	None	M
Date	Date	None	M

TABLE B.8: All fields for Research Group entity

Field Name	Data Type	Input format or Constraint	Mandatory (M) or Optional (O)
Group Name	Simple Text		M
Group Description	Multi-line Text		M
Members	Links to staff or post-graduate entities	None	M
Facilities	Links to facility entities	None	O

TABLE B.9: All fields for Staff entity

Field Name	Data Type	Input format or Constraint	Mandatory (M) or Optional (O)
Name	Simple Text	None	M
Position	Simple Text	None	M
Research Group	List (link to entity)	None	M
Photo	Image	Supported formats	O
Interests	Multi-line Text		M
Homepage	URL	None	O
Office	Simple Text	None	O
Email	Email format	None	M
Telephone	Simple Text	None	M
Biographical Description	Multi-line Text	None	O

TABLE B.10: All fields for Vacation Work entity

Field Name	Data Type	Input format or Constraint	Mandatory or Optional
Company	Simple Text	None	M
Description of Work	Multi-line Text	None	M
Contact Telephone Number	Simple Text	None	O
Contact Email	Email	Email validation	O
Website	URL	URL validation	O
Application form	Document	Attachment validation	O

References

- [1] R. Zakon, *Hobbes' Internet Timeline - the definitive ARPAnet and Internet history*, 2016. [Online]. Available: <http://zakon.org/robert/internet/timeline/#Growth> (visited on Aug. 13, 2016).
- [2] M. de Kunder, *The size of the World Wide Web (The Internet)*, 2008. [Online]. Available: <http://www.worldwidewebsite.com/> (visited on Feb. 1, 2014).
- [3] A Byron, J Robbins, and A Berry, *Using Drupal*, ser. O'Reilly Series. O'Reilly, 2009, ISBN: 9780596515805.
- [4] Oxford University Press, *CMS - Definition of word*, 2010. [Online]. Available: <http://oxforddictionaries.com/definition/english/CMS> (visited on Jan. 6, 2015).
- [5] W3Techs, *Usage statistics and market share of WordPress for websites*, 2016. [Online]. Available: <https://w3techs.com/technologies/details/cm-wordpress/all/all> (visited on May 26, 2016).
- [6] B Bonfield and L. S. Quinn, "Comparing Open Source CMSes: Joomla, Drupal and Plone," Idealware, Tech. Rep., 2007. [Online]. Available: <https://miab.tacticaltech.org/base.ngoinabox.org/files/basebox/Idealware/joomla/drupal/plone.pdf>.
- [7] J. Yuventi and J. Cortes, "Developing Rapidly Expandable Data-driven Software Systems Based on Multitier Architecture: Examination of the Development of Modules for Content Management Systems Used to Construct Web Portals and Virtual Communities," *2012 Ninth International Conference on Information Technology - New Generations*, pp. 331–334, 2012. DOI: 10.1109/ITNG.2012.56. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6209174>.

- [8] Drupal.org, *Drupal - Open Source CMS*, 2016. [Online]. Available: <http://drupal.org/> (visited on Sep. 11, 2016).
- [9] Wordpress, *WordPress.com Get a Free Blog Here*, 2016. [Online]. Available: <http://wordpress.com/> (visited on Nov. 3, 2016).
- [10] Joomla.org, *Joomla! The CMS trusted by millions for their websites*, 2016. [Online]. Available: <http://www.joomla.org/> (visited on Oct. 16, 2016).
- [11] C. Ayala, Ø. Hauge, R. Conradi, X. Franch, and J. Li, “Selection of third party software in Off-The-Shelf-based software development: an interview study with industrial practitioners,” *Journal of Systems and Software*, vol. 84, no. 4, pp. 620–637, 2011, ISSN: 01641212. DOI: 10.1016/j.jss.2010.10.019.
- [12] Drupal.org, *Distribution project — Drupal.org*, 2016. [Online]. Available: <https://www.drupal.org/project/project/distribution> (visited on Aug. 13, 2016).
- [13] Wordpress.org, *The Best WordPress Sites in the World*, 2016. [Online]. Available: <https://wordpress.org/showcase/> (visited on Aug. 13, 2016).
- [14] Joomla.org, *Home - Joomla! Community Showcase*, 2016. [Online]. Available: <http://showcase.joomla.org/sites.html> (visited on Aug. 13, 2016).
- [15] BuiltWith.com, *Ecommerce Usage Statistics Statistics for websites using Ecommerce technologies*, 2017. [Online]. Available: <https://trends.builtwith.com/shop> (visited on Feb. 6, 2017).
- [16] Tim Berners-Lee, *The original proposal of the WWW, HTMLized*, 1989. [Online]. Available: <http://www.w3.org/History/1989/proposal.html> (visited on Oct. 16, 2012).
- [17] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-down Approach*, ser. Always learning. Pearson, 2013, ISBN: 9780132856201.
- [18] S. McKeever, “Understanding Web content management systems: evolution, lifecycle and market,” *Industrial Management & Data Systems*, vol. 103, no. 9, pp. 686–692, 2003, ISSN: 0263-5577. DOI: 10.1108/02635570310506106.
- [19] B Boiko, *Content Management Bible*, ser. Bible Series. John Wiley & Sons, 2005, ISBN: 9780764583643.
- [20] Drupal.org, *About Drupal — drupal.org*, 2012. [Online]. Available: <http://drupal.org/about> (visited on Oct. 25, 2012).

- [21] S. R. G. Fraser, *Real World ASP.NET: Building a Content Management System*, ser. Real-World ASP.Net: Building a Content Management System. Apress, 2002, ISBN: 9781590590249.
- [22] DocForge, *Content management system - DocForge Programming Wiki*, 2011. [Online]. Available: <http://docforge.com/wiki/Content/management/system> (visited on Oct. 14, 2016).
- [23] CMS Critic, *Huge list of content management systems (List of CMS software)*, 2015. [Online]. Available: <https://www.cmscritic.com/dir/> (visited on Feb. 11, 2015).
- [24] Builtwith.com, *CMS technologies Web Usage Statistics*. [Online]. Available: <http://trends.builtwith.com/cms> (visited on Feb. 4, 2016).
- [25] R. Endsley, *2011 open source cms market share report*, 2011. [Online]. Available: <http://www.cmswire.com/cms/web-cms/2011-open-source-cms-market-share-report-wordpress-drupal-joomla-on-top-013679.php>.
- [26] R. Shreves, “2011 Open Source Market Share Report,” Water & Stone Consultancy, Bali, Indonesia, Tech. Rep., 2011. [Online]. Available: <http://www.waterandstone.com/downloads/2011OSCMSMarketShareReport.pdf> (visited on May 11, 2014).
- [27] Drupal.org, *Where is the Drupal Community? — Drupal.org*, 2017. [Online]. Available: <https://www.drupal.org/community> (visited on Feb. 18, 2017).
- [28] WordPress.org, *Get Involved*, 2017. [Online]. Available: <https://make.wordpress.org/> (visited on Feb. 18, 2017).
- [29] M. Butcher, *Drupal 7 Module Development: Create your own Drupal 7 modules from scratch*. Packt publishing, 2010, ISBN: 9781849511162.
- [30] J VanDyk and T Tomlinson, *Pro Drupal 7 Development*, ser. Books for professionals by professionals. Apress, 2011, ISBN: 9781430228394.
- [31] U. of the Witwatersrand, *School of Electrical And Information Engineering*, 2017. [Online]. Available: www.eie.wits.ac.za (visited on Feb. 20, 2017).
- [32] R. T. Douglass, M. Little, and J. W. Smith, *Building online communities with Drupal, phpBB, and WordPress*. Springer, 2006.

- [33] W. masters cafe, *CMS Software Comparison: WordPress vs. Joomla vs. Drupal*, 2011. [Online]. Available: <http://www.webmasterscafe.com/wordpress-vs-joomla-vs-drupal/> (visited on Nov. 9, 2015).
- [34] J. Wusteman, B. Eden, and A. Garza, "From OPAC to CMS: Drupal as an extensible library platform," *Library hi tech*, vol. 27, no. 2, pp. 252–267, 2009.
- [35] X. Cao and W. Yu, "Using content management system joomla! to build a website for research institute needs," in *Management and Service Science (MASS), 2010 International Conference on*, IEEE, 2010, pp. 1–3.
- [36] W. Powel and C. Gill, "Web content management systems in higher education," *Educause Quarterly*, vol. 26, no. 2, pp. 43–50, 2003.
- [37] S. D. Mooney and P. H. Baenziger, "Extensible open source content management systems and frameworks: a solution for many needs of a bioinformatics group," *Briefings in bioinformatics*, vol. 9, no. 1, pp. 69–74, 2008.
- [38] T. C. Chieu and Z. Liangzhao, "Service-oriented approach for implementing an extensible content management system," in *Proceedings - 2008 IEEE Congress on Services, SERVICES 2008*, vol. PART2, 2008, pp. 96–103, ISBN: 9780769532868. DOI: 10.1109/SERVICES-2.2008.14.
- [39] S. K. Patel, V. R. Rathod, and J. B. Prajapati, "Performance analysis of content management systems-joomla, drupal and wordpress," *International Journal of Computer Applications*, vol. 21, no. 4, pp. 39–43, 2011.
- [40] M. Meike, J. Sametinger, and A. Wiesauer, "Security in open source web content management systems," *IEEE Security & Privacy*, vol. 7, no. 4, 2009.
- [41] O. Aktunc, S. Dronavalli, and M. M. Tanik, "Rapid Prototyping of Digital Enterprises using Content Management Systems," in *IEEE SoutheastCon*, IEEE, 2008, pp. 231–235, ISBN: 9781424418848. DOI: <http://dx.doi.org/10.1109/SECON.2008.4494291>.
- [42] A. Brown and K. Wallnau, "A framework for evaluating software technology," *IEEE Software*, vol. 13, no. 5, pp. 39–49, 1996, ISSN: 07407459. DOI: 10.1109/52.536457. [Online]. Available: <http://ieeexplore.ieee.org/document/536457/>.

- [43] A. S. Jadhav and R. M. Sonar, "Evaluating and selecting software packages: A review," *Information and Software Technology*, vol. 51, no. 3, pp. 555–563, 2009, ISSN: 09505849. DOI: 10.1016/j.infsof.2008.09.003. [Online]. Available: <http://dx.doi.org/10.1016/j.infsof.2008.09.003>.
- [44] M Morisio, C. B. Seaman, V. R. Basili, A. T. Parra, S. E. Kraft, and S. E. Condon, "COTS-based software development: Processes and open issues," *Journal of Systems and Software*, vol. 61, no. 3, pp. 189–199, 2002, ISSN: 0164-1212. DOI: [http://dx.doi.org/10.1016/S0164-1212\(01\)00147-9](http://dx.doi.org/10.1016/S0164-1212(01)00147-9). [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0164121201001479>.
- [45] M Fowler, *Patterns of Enterprise Application Architecture*, ser. Addison-Wesley Signature Series (Fowler). Pearson Education, 2012, ISBN: 9780133065213.
- [46] K Qian, *Software Architecture and Design Illuminated*, ser. Jones and Bartlett illuminated series. Jones & Bartlett Learning, 2010, ISBN: 9780763754204.
- [47] E. Evans, *Domain-driven design: Tackling complexity in the heart of software*. Addison-Wesley, 2004, ISBN: 9780321125217.
- [48] R. T. Futrell, D. F. Shafer, and L Shafer, *Quality Software Project Management*, ser. Software Quality Institute series. Prentice Hall, 2002, ISBN: 9780130912978.
- [49] W. S. Humphrey, *Introduction to the Personal Software Process*, ser. SEI series in software engineering. Addison-Wesley, 1997, ISBN: 9780201548099.
- [50] W Frakes and C Terry, "Software reuse: metrics and models," *ACM Computing Surveys (CSUR)*, vol. 28, no. 2, pp. 415–435, 1996.
- [51] J. E. Matson, B. E. Barrett, and J. M. Mellichamp, "Software development cost estimation using function points," *IEEE Transactions on Software Engineering*, vol. 20, no. 4, pp. 275–287, 1994.
- [52] P. P.-S. Chen, "The Entity-relationship Model: Toward a Unified View of Data," *ACM Trans. Database Syst.*, vol. 1, no. 1, pp. 9–36, 1976, ISSN: 0362-5915. DOI: 10.1145/320434.320440. [Online]. Available: <http://doi.acm.org/10.1145/320434.320440>.
- [53] R Sperko, *Java Persistence for Relational Databases*. Apress, 2008, ISBN: 9781430208167.
- [54] CKEditor.org, *CKEditor Interface*, 2011. [Online]. Available: <http://docs.cksource.com/CKEditor/3.x/Users/Guide/Interface> (visited on Dec. 2, 2016).

- [55] Google.com, “Search Engine Optimization Starter Guide,” Google inc., Tech. Rep., 2010. [Online]. Available: <http://www.google.com/webmasters/docs/search-engine-optimization-starter-guide.pdf> (visited on May 11, 2016).
- [56] —, *About Sitemaps*, 2012. [Online]. Available: <http://support.google.com/webmasters/bin/answer.py?hl=en&answer=156184>.
- [57] R Patton, *Software Testing*. Sams Pub., 2006, ISBN: 9780672327988.
- [58] D. F. Galletta, R. Henry, S. McCoy, and P. Polak, “Web site delays: How tolerant are users?” *Journal of the Association for Information Systems*, vol. 5, no. 1, p. 1, 2004.
- [59] J. Nielsen, *Website Response Times*, 2010. [Online]. Available: <https://www.nngroup.com/articles/website-response-times/> (visited on Mar. 2, 2017).
- [60] TechRepublic.com, *Tips for optimizing your web images*. [Online]. Available: <http://www.techrepublic.com/blog/web-designer/tips-for-optimizing-your-web-images/> (visited on Mar. 4, 2017).
- [61] S. K. Patel, V. R. Rathod, and J. B. Prajapati, “Comparative analysis of web security in open source content management system,” in *2013 International Conference on Intelligent Systems and Signal Processing (ISSP)*, 2013, pp. 344–349. DOI: 10.1109/ISSP.2013.6526932.
- [62] Drupal.org, *Securing your site*, 2016. [Online]. Available: <https://www.drupal.org/security/secure-configuration> (visited on Mar. 2, 2017).
- [63] T. S. Inc., *Website Security*, 2017. [Online]. Available: <https://www.tinfoilsecurity.com/> (visited on Mar. 2, 2017).
- [64] Drupal.org, *Installing Drupal 7*, 2017. [Online]. Available: <https://www.drupal.org/docs/7/install> (visited on Mar. 1, 2017).
- [65] DigitalOcean, *Drupal CMS Server One-Click Install & Deploy to Cloud — DigitalOcean*, 2017. [Online]. Available: <https://www.digitalocean.com/products/one-click-apps/drupal/> (visited on Mar. 1, 2017).
- [66] GoDaddy Inc., *Drupal Hosting — Fast, One Click Cloud Install - GoDaddy ZA*, 2017. [Online]. Available: <https://za.godaddy.com/pro/one-click-installation/drupal> (visited on Mar. 1, 2017).

-
- [67] Drupal.org, *Crashed Drupal Sessions Table*. [Online]. Available: <https://www.drupal.org/node/833762> (visited on Feb. 4, 2016).
- [68] B Melancon, A Micka, A Scavarda, B Doherty, B Somers, J Rodriguez, K Negyesi, M Weitzman, R Scholten, R Szrama, and Others, *The Definitive Guide to Drupal 7*, ser. Expert's voice in Web development. Apress, 2011, ISBN: 9781430231363.
- [69] L. Clark, *How Drupal works: Architects overview*, 2010. [Online]. Available: <http://lin-clark.com/blog/how-drupal-works-architects-overview/25E2/2580/2594notes> (visited on Oct. 31, 2014).