Projects in Computing and Information Systems

A Student's Guide

Third Edition

Christian W. Dawson

ALWAYS LEARNING

PEARSON

Projects in Computing and Information Systems

Projects in Computing and Information Systems

A Student's Guide

Third Edition

CHRISTIAN W. DAWSON

PEARSON

Harlow, England • London • New York • Boston • San Francisco • Toronto • Sydney
Auckland • Singapore • Hong Kong • Tokyo • Seoul • Taipei • New Delhi
Cape Town • São Paulo • Mexico City • Madrid • Amsterdam • Munich • Paris • Milan

Pearson Education Limited
Edinburgh Gate
Harlow CM20 2JE
United Kingdom
Tel: +44 (0)1279 623623
Web: www.pearson.com/uk

First published 2005 (print)
Second edition 2009 (print)
Third edition published 2015 (print and electronic)

ISBN: 978–1-292–07346–0 (print)
978–1-292- 08112–0 (PDF)
978–1-292–08111–3 (eText)

British Library Cataloguing-in-Publication Data
A catalogue record for the print edition is available from the British Library

Library of Congress Cataloging-in-Publication Data
Dawson, Christian W.
Projects in computing and information systems : a student's guide/Christian W. Dawson.
— Third edition.
pages cm
Includes bibliographical references and index.
ISBN 978-1-292-07346-0
1. Electronic data processing. 2. Information technology. I. Title.
QA76.D3333326 2015
004—dc23

2014042818

A catalog record for the print edition is available from the Library of Congress
10 9 8 7 6 5 4 3 2 1
19 18 17 16 15

Cover image © Getty Images
Print edition Typeset in 9/11pt Galliard by 71
Printed in Malaysia

For Jacob and Ben

Contents

Preface

Projects are a major component of virtually all undergraduate and postgraduate computing and information science courses within universities. They require students to draw on a number of separate but highly important skills: surveying literature, report writing, developing and documenting software, presentational skills, time management, project management skills and so on. For students to excel in all of these areas is a major accomplishment, yet it is something that academic institutions have come to expect as part of the independent learning process.

Although there are books available that cover *some* of these topics in great detail, there are none that draw **all** these skills together and which are aimed specifically at students on computing and information systems courses of one kind or another. This text fills this gap and provides a foundation in the skills both undergraduate and postgraduate students require to complete their projects successfully.

This book is structured in a chronological fashion so that the main stages through which projects progress are discussed in sequence. It is split into five main sections.

1. **The background.** This section provides a general introduction to projects, the different degree structures that are in place and the stakeholders involved. It also provides a useful introduction to research in the context of computing projects.
2. **Setting your project's foundation.** This section covers the skills you will need during the initial stages of your computing project. It covers topics such as how to choose a project, how to write a project proposal, and how to plan your project.
3. **Conducting your project.** This section covers the skills you will need while you are actually working on your project – from doing your literature survey to managing your time and any information and data that you collect, as well as how to liaise effectively with your supervisor. It also includes a chapter on software development for those undertaking projects of this nature.

4. **Presenting your project.** The final stage of your project is to present it as a written report and, possibly, an oral presentation. This section will cover the skills you will need to present your project in the best light and to the best of your abilities.
5. **The future.** The book concludes with some valuable information on how your project might be assessed, how you can take your project further in the future and how you might consider publishing your work.

The *mortarboard* symbol emphasises parts of the book specifically aimed at research degrees (PhD, DPhil, etc.).

Acknowledgements

The author and publisher would like to express their thanks to the following reviewers for their invaluable feedback throughout the development of this book:

Jackie Archibald, University of Abertay
Xiaodong Liu, Napier University
Diane Richardson, University of Bedfordshire
Clive Rosen, University of Derby

Publisher's acknowledgements

We are grateful to the following for permission to reproduce copyright material:

Figures

Figures 6.13, 6.14 adapted from Ould, M. (1999) *Managing Software Quality and Risk*, © 1999 John Wiley & Sons Ltd. Reproduced with permission of John Wiley & Sons Ltd.; Figure 9.10 from Yanning Yang, Loughborough University; Figure 9.11 from Rachael Lindsay, Loughborough University; Figure 9.12 from Martin Sykora, Loughborough University.

Tables

Table 7.3 adapted from *Time Management: the essential guide to thinking and working smarter*, Marshall Editions Ltd (Jones, K. 1998), © Copyright Marshall Editions.

Text

Epigraph on page 17 from *How to get a PhD*: *a handbook for students and their supervisors*, 5th edn, Open University Press (Phillips, E.M. and Pugh, D.S. 2010) p. 56, McGraw-Hill Education; Text on pages 33–35 adapted from *How to Research*, 4th edn, Open University Press (Blaxter, L. Hughes, C. and Tight, M. 2010) p. 203, McGraw-Hill Education; Example on pages 111–12 adapted from Dawson, C.W. and Wilby, R. (1998) An artificial neural network approach to rainfall-runoff modelling, *Hydrological Sciences Journal*, 43(1), pp. 47–66 reprinted by permission of the publisher (Taylor & Francis Ltd, http://www.tandfonline.com)

In some instances we have been unable to trace the owners of copyright material, and we would appreciate any information that would enable us to do so.

SECTION 1

The background

CHAPTER

1

Introduction

Aims:

To introduce academic computing projects and the structure of this book.

Learning objectives:

When you have completed this chapter, you should be able to:

- Understand what projects are.
- Understand the different types of academic projects in computing and information sciences.
- Understand different degree structures and project requirements.
- Describe the roles different people have in academic projects.
- Understand how this book is arranged.

1.1 Introduction

Pursuing a project within academia is not the same as performing a project within industry. As a student on a computing degree course of one kind or another, you will be expected to look at things much more critically and more deeply than you would elsewhere. In industry, for example, your line manager might ask you to develop a piece of software to solve a particular problem or improve productivity in a particular area – a database, a production control system or whatever. You could write this program well and install it within a few weeks or months and everyone would be satisfied. However, although this program might be perfectly adequate and work very well in practice, this project would be lacking *academically*.

Why is this the case? Academic projects should provide evidence of a much deeper understanding of what you are doing. They require some form of justification and contextualisation. You are not expected to do merely what you are told to do, but you are expected to develop your own thoughts, arguments, ideas and concepts. You are expected to question things and look at things in new ways and from new angles. Merely 'turning the handle' or doing what you are told does not lead to intellectual discovery and contributions to world thinking. Importantly, as a degree student you are expected to **think.** This 'deeper' understanding of situations, problems and events is supported by your *research* skills – skills that are vitally important within academic projects.

Academic projects are usually a critical component of your degree course. Sometimes they make up a significant component of your final year (for example, 30% or more), and sometimes, particularly at postgraduate level, they may represent *all* of your degree. There are a number of reasons why universities include project work as part of their courses.

- **Assessment across a number of disciplines simultaneously.** Your project will require you to apply things you have learnt from many different areas of your course – both technical and personal skills. The project will provide evidence of how much you have developed across a wide range of disciplines. It will also show your ability to draw together your knowledge and apply what you have learnt. This might be the first time that the importance of apparently different skills taught on your course may become clear.
- **Allows you to develop new skills.** The project will also enable you to develop skills you might not have covered explicitly on your course so far. These new skills might be technical (learning a new programming language, development method, design technique, research, etc.) and personal (time management, discipline, communication skills, report writing, etc.).
- **Work independently.** Your project might be the first time that you have had to work mainly on your own on a project that is primarily your own work, ideas and responsibility.
- **Make a contribution.** A project will allow you to make some form of contribution. Previously you might have been doing directed coursework, examinations, etc. The project will allow you to produce something that may be used by or benefit others.

This book aims to help you with this critical component of your course – be it at undergraduate or postgraduate level.

1.2 What are (computing) projects?

1.2.1 Introduction

Projects can be defined as 'something which has a beginning and an end' (Barnes, 1989 cited by Turner, 1993: 4). Unfortunately, this rather broad definition of projects does not encapsulate their underlying purpose, which is to bring about some form of beneficial change. This change takes you from an existing situation to a desired situation sometime in the future. This can be represented by the *Meliorist Model* shown in Figure 1.1. In this figure a project is represented by a set of actions that you perform. A project thus enables you to move from one situation to another. Your movement towards the desired situation might stem from dissatisfaction with your current situation, a lure towards a situation which appears more satisfactory, or some combination of the two.

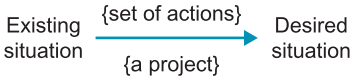

Figure 1.1 The Meliorist Model

The desirable situation in this case represents some form of contribution to knowledge – perhaps representing the development of a new tool, technique, discovery and so on. The term 'contribution' in this context necessarily implies the uniqueness of the project and novelty of its outcomes.

So far projects have been identified as having a beginning and an end (i.e., they occur within a specified **time frame**) – with a **purpose** being to bring about a beneficial change by making some kind of contribution. Another important aspect of projects that must be discussed is that they are made up of a series of **considered** activities. In other words, projects are broken down into a sequence of **planned** activities that are controlled as the project progresses – they do not simply occur in an *ad hoc* manner. This aspect of projects – project planning and risk management– is looked at in detail in Chapter 4. We should also consider the fact that projects consume a number of **resources** to achieve their purpose. What resources are consumed in your project is introduced in Chapter 4. How these resources are managed is discussed in Chapter 7.

Computing projects come in all different shapes and sizes, as the field they are drawn from is immense. However, these days it is more widely recognised, within academic institutions, that computing projects need to do more than develop a piece of software. The project that you pursue must involve an element of research, it must justify its context and evaluate and discuss its results. Merely developing a tool or algorithm with no evaluation or contextualisation may well be acceptable in industry, where commercial solutions are required. However, within the academic world, this is not the case and, depending on the nature of your project, it will have to contain an element of research to a greater or lesser extent.

Berndtsson *et al.* (2008) point out that the nature of computer science and information systems means that projects are drawn from both the 'hard' sciences (natural science) and the 'soft' sciences (social sciences). Consequently, projects cover a vast range of topics, from highly technical software development projects to (equally difficult) case studies within information science. Figure 1.2 (adapted from Dawson, 2004) shows the extent of the computer science and information science field. At the left-hand side of this scale are the theoretical areas of computer science. These encompass areas such as mathematics, logic, formal methods, artificial intelligence and so on. Moving towards the centre of the scale we find practice-based computing. This focuses less on theory and more on the development of software systems – for example, software engineering, software project management, design, development processes, requirements capture and so on. At the right-hand side of the scale are the softer issues in the field. These are concerned with the application, use, influence and impact that computers

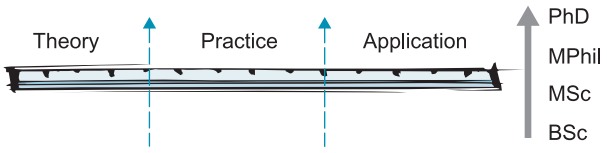

Figure 1.2 The landscape of computing (adapted from Dawson, 2004)

and information technology have on organisations and society at large. This side is also concerned with how organisations are structured and operate, how data are stored and processed, knowledge management and how information is disseminated throughout organisations. This might involve systems beyond software and hardware – for example, paper-based systems, processes, procedures and so on. Figure 1.2 also shows the different levels of project that this book covers on the vertical scale – from undergraduate projects to doctoral degrees. These levels are discussed in Section 1.3.

In terms of university degree courses, you will *probably* find that courses entitled 'Computer Science' or 'Artificial Intelligence' tend to fall more towards the left-hand side of this scale. 'Software Engineering', 'Computing' and 'E-business' courses probably fall more towards the centre (with 'Software Engineering' to the left and 'E-business' to the right). Courses entitled 'Information Science', 'Information Technology', 'Business and Information Technology', 'Business Information Systems' and 'Information Systems' will fall more towards the right-hand side. Some courses might fall anywhere along the scale depending on their content – for example, 'Multimedia' and 'Computer Studies' can mean different things to different institutions.

This list is not intended to be exhaustive and you may find that your course falls somewhere else along this line. You should, however, have some idea of where your course lies on this scale as this will influence the type of project that is appropriate for you to undertake. This book will address the issues surrounding all of these areas.

The computing project that you embark upon gives **you** an opportunity to make your **own** contribution. There is little point in doing a project that merely regurgitates the work of others. Your own thoughts, ideas and developments **are** important and these are the things that people reading your report are interested in. It is through your project that you will develop not only your own skills but also the ideas and work of others. The level of contribution that undergraduate and postgraduate projects make is looked at in more detail in Section 5.1.

The following section introduces the different kinds of project that you are likely to encounter within the field of computing. In each of these cases we identify how these projects make some kind of academic contribution. They do not merely follow a simplistic project process to develop a product at the end of the day.

1.2.2 Computing project types

This section outlines five categories of computing projects. These categories are not intended to be discrete and you may well find that your own project falls into two or even more of these classes (or it perhaps falls distinctly into one category but draws on approaches that are identified in others). In addition, the nature of your project will have an effect on the methods you will use to tackle it. The research methods that you might employ within your project are discussed in Chapter 2.

- **Research-based.** 'Many good dissertations do no more than review systematically and impose some structure on, a field of interest' (Sharp *et al.*, 2002: 27). A research-based project involves a thorough investigation of a particular area; improving your understanding of that area, identifying strengths and weaknesses within the field, discussing how the field has evolved, and acknowledging areas suitable for further development and investigation (identifying gaps). This kind of project will involve some form of literature search and review and would be suitable for taught bachelor's or taught master's courses.

So far, this definition of research-based projects has been 'backward looking' rather than 'forward looking' (Cornford and Smithson, 2006: 71). A research-based project may well have to do more than establish the field of study. For example, having established the field (backward looking), a doctoral degree (a PhD, for example) would then be expected to contribute to that field (forward looking). This contribution might be achieved by addressing a research question, developing something new, solving a problem and so on. These additional steps are addressed in the following sections in which other project types are defined.

- **Development.** This category includes the development not only of software and hardware systems but also of process models, methods, algorithms, theories, designs, requirement specifications and other interim documents. Examples of software development projects are database systems, apps for phones and other portable devices, multimedia systems, information systems and web-based systems. For some developments (notably software) you will be required to include requirements documentation, designs, analyses and fully documented test results, along with user manuals or guides. The type of development you undertake will also affect the issues and problems you might face. For example, a web development project may have security issues, interface design issues, technical issues (perhaps related to the content management system you choose to use) and so on. A stand-alone system will have issues surrounding, for example, portability and efficiency.

  Depending on the nature of your course the focus for a development project may vary. For example, for software engineering courses, emphasis may be placed on the development and evaluation of a piece of software, following particular process models that generate interim evaluatory documentation. Information systems courses may require you to focus more on the development of broader systems using 4GLs, CASE tools and/or database systems. In this case evaluation of HCI (human–computer interaction), customer issues, requirements capture problems and the impact of the implemented system and working practices may be more your focus.

  Whichever kind of development project you tackle it is unlikely that the development of a product would be acceptable on its own. You would normally be expected to include a critical evaluation of the product as well as the development process used. Critical evaluation emphasises the distinction between the academic qualities of your work and technical ability alone. Figure 1.3 illustrates this point and contrasts a student project and an industrial-based project that are both attempting to develop a software system to meet a user's need. Although the student project goes someway to meeting the user's needs for a system, there is much more to the project than that (and hence the user's need for a system might be only partially fulfilled). In contrast,

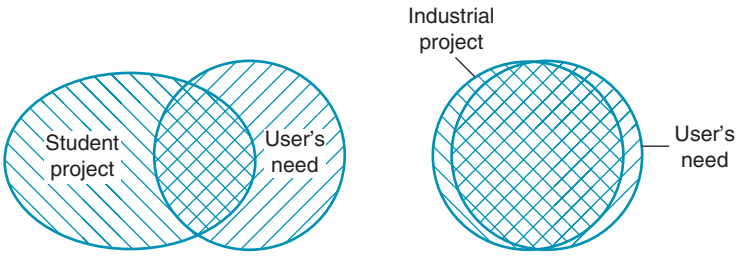

Figure 1.3 Comparison of student development project and industrial development project

the industrial-based project is primarily focused on solving the user's problem and little else. Consequently the industrial project develops a system that is as close as possible to meeting the requirement of the user (although it is never perfect), whereas the student project does not.

- **Evaluation.** This category encompasses all projects that involve some form of evaluation as their main focus. For example, such a project might involve comparing several approaches to a particular problem; evaluating two or more programming languages (applied in different contexts or to different problems); analysing an implementation process within a particular industry; assessing different user interfaces; appraising a particular concept; assessing alternative and new technological approaches to a problem; analysing development methodologies to a problem; and so on. Projects in this category may well include case studies as a vehicle for evaluating the issue under consideration.
- **Industry-based.** An industry-based project involves solving a problem within either an organisation or another university department. Industry-based projects might be any of the other kinds of projects identified in this section. The difference in this case is that you are undertaking the project for an actual client, which carries with it a number of benefits as well as drawbacks. The pitfalls and benefits of choosing an industry-based project are discussed in more detail in Section 3.2. The most important point is that the sponsor does not 'hijack' the project – that is, force it into a direction that the company wishes it to go, regardless of whether it is suitable for your academic work or your course. You will probably find that an action research method is employed in this kind of project (discussed in Section 2.4).
- **Problem solving.** A problem-solving project can involve developing a new technique to solve a problem, improving the efficiency of existing approaches or evaluating different approaches or theories in different situations. It might also involve applying an existing problem-solving technique or theory to a new area. In these cases, some form of evaluation would be expected: for example, did your new approach work well or did you discover reasons why it was unsuitable for problems of this nature? Why does one approach or theory work better in some situations than in others?

1.2.3 Examples of projects in different areas of computing and information systems

In this section we present some examples of the types of projects students might undertake in the different areas within computing and information sciences. The list is by no means exhaustive and there will be some argument as to the different fields identified (you might think of other areas not listed here or feel that some should not be here at all). However, the list should provide some ideas on the types of projects you might be interested in and the sort of area you might want to work in. The areas are presented alphabetically.

- **Algorithms and data structures.** Working on improvements to existing algorithms and data structures; evaluating algorithms and data structures in different contexts.
- **Applied computer science.** Application of computer science in other fields. Study of computer science and information science in different problem domains; for example, how computer science is used in the medical sciences or how information science is used in local government.

- **Artificial intelligence (AI).** Development of new AI techniques or making improvements to existing techniques; applying existing techniques to new problems; evaluating AI techniques in different problem domains.
- **Computer architectures and hardware.** Linking computers to hardware devices and developing systems to support and manage those devices; for example, linking computers to engine management systems; measuring the performance of different architectural components and their interactions; exploring new ways of structuring and linking hardware; discovering new ways of structuring computer architecture.
- **Databases.** Design and development of a database to solve a problem; evaluating or developing different ways of designing database structures; assessing database structures; developing ways of measuring databases; researching commit protocols and improvements to them; developing distributed databases and efficient means of designing them.
- **Formal methods.** Producing specifications and verification of processes or systems; reverse engineering formal specifications from existing systems; developing formal specification languages or adaptations to existing ones.
- **Graphics and visualisation.** Working with subdivision algorithms to build an image for geometric modelling; developing or using algorithms to analyse images.
- **Human–computer interaction (HCI).** Evaluating HCI in different systems for different purposes; designing system interfaces for specific purposes.
- **Image processing, vision, pattern recognition.** Image modification; repairing negatives or photographs; image compression algorithms; handwriting recognition.
- **Information systems.** Research projects into how a company uses information; knowledge management; use of IT in different areas/organisations; information on social media; privacy; cookies and cookie policies; Data Protection Act.
- **Networking.** Measurement of network performance; protocol analysis; network security; security evaluation of protocols; designing and evaluating network architectures; wireless networks.
- **Security and cryptography.** Evaluating the effectiveness of different security protocols; implementing and evaluating public cryptography algorithms; developing new cryptography algorithms; developing pseudo random number generators for cryptography algorithms.
- **Software engineering.** Evaluating process models in different projects; developing new processes; developing new techniques for undertaking different stages of the development life cycle; finding ways to analyse software structure; evaluating maintenance issues of software; measuring software in new ways.
- **Theoretical computer science.** Implementing an algorithm; solving basic problems; designing algorithms; proving theorems and establishing theories.

1.2.4 Programming in computing projects

Although you are on a computing course of one kind or another, it is not necessarily the case that you will be expected to write a program. As noted earlier, the broad field of computing encompasses many topics such as information systems, software engineering, knowledge engineering, HCI, data communications, networks and computer systems