

## University of Southampton Research Repository ePrints Soton

Copyright © and Moral Rights for this thesis are retained by the author and/or other copyright owners. A copy can be downloaded for personal non-commercial research or study, without prior permission or charge. This thesis cannot be reproduced or quoted extensively from without first obtaining permission in writing from the copyright holder/s. The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the copyright holders.

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given e.g.

AUTHOR (year of submission) "Full thesis title", University of Southampton, name of the University School or Department, PhD Thesis, pagination



Faculty of Engineering and the Environment  
University of Southampton  
United Kingdom

# Research data management

Thesis submitted for the degree of  
Doctor of Philosophy  
by Mark Scott

May 2014



**UNIVERSITY OF SOUTHAMPTON**

**ABSTRACT**

**FACULTY OF ENGINEERING AND THE ENVIRONMENT**

**Computational Engineering and Design**

**Doctor of Philosophy**

**RESEARCH DATA MANAGEMENT**

**by Mark Scott**

Scientists within the materials engineering community produce a wide variety of data, ranging from large 3D volume densitometry files (voxel) generated by microfocus computer tomography ( $\mu$ CT) to simple text files containing results from tensile tests. Increasingly they need to share this data as part of international collaborations. The design of a suitable database schema and the architecture of a flexible system that can cope with the varying information is a continuing problem in the management of heterogeneous data.

We discuss the issues with managing such varying data, and present a model flexible enough to meet users' diverse requirements. Metadata is held using a database and its design allows users to control their own data structures. Data is held in a file store which, in combination with the metadata, gives huge flexibility and means the model is limited only by the file system. Using examples from materials engineering and medicine we illustrate how the model can be applied. We will also discuss how this data model can be used to support an institutional document repository, showing how data can be published in a remote data repository at the same time as a publication is deposited in a document repository.

Finally, we present educational material used to introduce the concepts of research data management. Educating students about the challenges and opportunities of data management is a key part of the solution and helps the researchers of the future to start to think about the relevant issues early on in their careers. We have compiled a set of case studies to show the similarities and differences in data between disciplines, and produced documentation for students containing the case studies and an introduction to the data lifecycle and other data management practices.

Managing in-use data and metadata is just as important to users as published data. Appropriate education of users and a data staging repository with a flexible and extensible data model supports this without precluding the ability to publish the data at a later date.

## **Research data management**

© Mark Scott, University of Southampton 2014

Copyright and Moral Rights for this thesis are retained by the author and/or other copyright owners. A copy can be downloaded for personal non-commercial research or study, without prior permission or charge. This thesis cannot be reproduced or quoted extensively from without first obtaining permission in writing from the copyright holder/s. The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the copyright holders.

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given, i.e. Mark Scott. 2014. 'Research data management'. PhD diss., University of Southampton.

To Yeyang and my wonderful family who kept me going during the low times and celebrated with me during the high times, and to my father who finally lost his ten year battle with malignant mesothelioma on his 75th birthday just before the viva and so didn't quite see the end of this journey.



*A rose blooms and then fades,  
but the beauty and the fragrance  
are remembered always.*  
(Eddings 1987)



# Contents

---

<i>List of figures</i> . . . . .	xiii
<i>List of tables</i> . . . . .	xvii
<i>Declaration</i> . . . . .	xix
<i>Acknowledgements</i> . . . . .	xxi

<b>Chapter 1 Introduction</b>	<b>1</b>
1.1 A historical warning . . . . .	1
1.2 ‘Standing on the shoulders of giants’ . . . . .	2
1.3 Data sharing . . . . .	3
1.4 Research data management . . . . .	3
1.5 Data capture . . . . .	4
1.6 Materials engineering data . . . . .	5
1.7 Thesis outline . . . . .	6
1.7.1 Main research questions and thesis objectives . . . . .	6
1.7.2 Structure . . . . .	7
 <b>Chapter 2 Computing data and associated technologies</b>	 <b>11</b>
2.1 Introduction . . . . .	11
2.2 Terminology . . . . .	11
2.3 Computing data . . . . .	12
2.3.1 File data . . . . .	13
2.3.2 Metadata . . . . .	18
2.4 Data storage . . . . .	19
2.4.1 Storage hardware . . . . .	19
2.4.2 Local file systems . . . . .	20
2.4.3 Network file systems . . . . .	22
2.4.4 Distributed file systems . . . . .	22
2.4.5 Cloud technologies . . . . .	22
2.4.6 Data transfer . . . . .	24



2.5	User requirements collection . . . . .	24
2.5.1	User requirements report and questionnaire . . . . .	25
2.5.2	Synchrotron data interview . . . . .	28
2.5.3	MatDB schema interview . . . . .	30
2.6	Materials engineering data . . . . .	32
2.6.1	Tensile test . . . . .	32
2.6.2	Long crack growth fatigue test . . . . .	33
2.6.3	Plain bend bar testing . . . . .	33
2.6.4	Fractography . . . . .	34
2.6.5	Microfocus X-ray computed tomography . . . . .	34
2.6.6	Material information . . . . .	35
2.6.7	Relationships between tests . . . . .	35
2.6.8	Summary . . . . .	37
2.7	Database technologies . . . . .	37
2.7.1	Relational database systems . . . . .	37
2.7.2	Normalisation . . . . .	37
2.7.3	Entity-Attribute-Value . . . . .	37
2.7.4	Federated databases and GaianDB . . . . .	40
2.8	Structured storage . . . . .	41
2.8.1	The Content Repository for Java Technology API specification . . . . .	42
2.9	Document repositories . . . . .	42
2.9.1	Institutional document repositories . . . . .	42
2.9.2	Microsoft SharePoint . . . . .	43
2.10	Protocols and frameworks . . . . .	44
2.10.1	RDF (Resource Description Framework) . . . . .	44
2.10.2	Atom Publishing Protocol and SWORD . . . . .	44
2.10.3	OAI-ORE (Open Archives Initiative Object Reuse and Exchange) . . . . .	44
2.10.4	OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting) . . . . .	45
2.11	Accurate data capture . . . . .	45
2.11.1	Semantic web . . . . .	46
2.11.2	Discussion . . . . .	46
2.12	Data management projects . . . . .	47
2.12.1	The Materials Atlas (materials data repository) . . . . .	48
2.12.2	Human Genome Project . . . . .	49
2.12.3	Large Hadron Collider . . . . .	49

2.13	Summary	50
<b>Chapter 3</b>	<b>Data publishing with EPrints</b>	<b>53</b>
3.1	Introduction	53
3.2	Architecture	54
3.2.1	Early MDC prototype	55
3.2.2	Service layer	55
3.2.3	EPrints plug-in	58
3.3	Results	58
3.4	Summary	61
<b>Chapter 4</b>	<b>A model for managing materials data</b>	<b>63</b>
4.1	Introduction	63
4.2	Architecture	64
4.2.1	File system	65
4.2.2	Metadata database	66
4.2.3	Logic layer	68
4.2.4	File system monitor	69
4.2.5	Interface	73
4.3	Materials engineering use cases	76
4.3.1	Material information	76
4.3.2	Tensile test	79
4.3.3	Long crack growth fatigue test	79
4.3.4	Microfocus computed tomography data	81
4.4	Reliability tests	82
4.4.1	Tests of the synchronisation service	82
4.4.2	Testing the metadata database	82
4.5	Performance tests	84
4.6	Working with data in the system	84
4.6.1	Searching using the interface tools	85
4.6.2	Data set suggestions	85
4.6.3	Data access API	85
4.6.4	SQL queries	86
4.6.5	API example queries	86

4.7	Discussion . . . . .	90
4.7.1	Feedback from user testing . . . . .	91
4.7.2	Metadata and data storage . . . . .	92
4.7.3	Materials Data Centre interface . . . . .	93
4.8	Summary . . . . .	93
<b>Chapter 5</b>	<b>A model for managing heterogeneous data</b>	<b>95</b>
5.1	Introduction . . . . .	95
5.2	Improvements since MDC . . . . .	95
5.2.1	Final metadata database schema . . . . .	96
5.2.2	Per-user and per-group data sets . . . . .	96
5.2.3	User driven data set security . . . . .	98
5.2.4	Data set details page redirection . . . . .	98
5.2.5	Metadata reordering . . . . .	98
5.2.6	Plug-ins . . . . .	99
5.2.7	Interface improvements . . . . .	102
5.2.8	RDF representation of data and metadata . . . . .	102
5.3	Use cases . . . . .	103
5.3.1	Human genetics use case . . . . .	103
5.3.2	Finding data using SPARQL . . . . .	105
5.3.3	EP2DC revisited . . . . .	107
5.4	Discussion . . . . .	109
5.4.1	Straightforward data deposit process . . . . .	109
5.4.2	Metadata and metadata tools . . . . .	109
5.4.3	Data set discoverability . . . . .	110
5.4.4	Data set tools adding value to users . . . . .	111
5.4.5	Cross-discipline support . . . . .	111
5.4.6	Future work . . . . .	112
5.5	Summary . . . . .	112
<b>Chapter 6</b>	<b>Research data management education</b>	<b>115</b>
6.1	Introduction . . . . .	115
6.2	About the <i>Introducing Research Data</i> guide . . . . .	116
6.2.1	Guide Part I: ‘Five Ways To Think About Research Data’ . . . . .	116
6.2.2	Guide Part II: ‘Case Studies’ . . . . .	116
6.2.3	Guide Part III: ‘Data Management Practices’ . . . . .	117

6.3	The <i>Introducing Research Data</i> guide . . . . .	118
6.4	Feedback received . . . . .	148
6.4.1	Faculty of Engineering Sciences and the Environment . . . . .	148
6.4.2	WebScience Doctoral Training Centre . . . . .	148
6.4.3	Gradbook training event . . . . .	148
6.5	Evaluation . . . . .	149
6.6	Summary . . . . .	149
<b>Chapter 7</b>	<b>Summary and outlook</b>	<b>151</b>
7.1	Discussion of objectives . . . . .	151
7.2	Discussion of research questions . . . . .	153
7.3	Outlook . . . . .	155
<b>Appendix A</b>	<b>User requirements report questionnaire results</b>	<b>157</b>
<b>Appendix B</b>	<b>MatDB schema user interviews</b>	<b>161</b>
<b>Appendix C</b>	<b>User feedback</b>	<b>169</b>
C.1	Email containing user feedback . . . . .	169
C.2	Email containing summary of meeting with a user and their feedback	170
C.3	Email containing summary of meeting with a user and their feedback	170
C.4	Email containing user feedback and summary of experience . . . . .	171
<i>Bibliography</i>	. . . . .	173



# List of figures

---

1.1	The Digital Curation Centre's data lifecycle (Higgins 2008) . . . . .	4
2.1	Fatigue test configuration . . . . .	26
2.2	Examples of the figures given in Appendix B, used in the MatDB XML schema interviews . . . . .	30
2.3	Material information sheet, currently used at the University of Southampton for recording a material's composition and for describing the properties of the material. Also shows images of the material's microstructure . . . . .	36
2.4	Relationships between the different types of test data. The Materials Data Centre needs to be able to represent this . . . . .	36
3.1	EP2DC Deposit Workflow. The 'EP2DC Data' step is handled by the EP2DC EPrints plug-in . . . . .	54
3.2	Components in the EP2DC architecture and an illustration of the conversation between EPrints and the remote data centre when a data file is deposited . . . . .	54
3.3	The mdcschemas document library used to hold XML schemas for XML validation . . . . .	57
3.4	Metadata entry for a data file being deposited to an external data centre using the EP2DC plug-in . . . . .	59
3.5	Containers are shown as folders in the MDC. GUIDs are used as the names of the folders . . . . .	59
3.6	This file was deposited here by EPrints inside a container. This is how it is displayed in the MDC prototype's interface . . . . .	60
3.7	How EPrints displays the publication and the information given to it by the MDC . . . . .	60
4.1	Data flow in data management model . . . . .	64
4.2	Metadata database tables in data management model . . . . .	66
4.3	Screenshot showing list of a user's data sets, including a thumbnail if present and data set summary . . . . .	73
4.4	Data set details page – (1) summary section, showing (a) data set plug-ins and (b) a data set thumbnail . . . . .	74

4.5	Data set details page – metadata sections: (1) metadata and (2) metadata tools. Shows (a) nested metadata, (b) clickable metadata keywords and (c) data set search boxes . . . . .	74
4.6	Data set details page – (1) data files section, showing (a) data file plug-ins . . . . .	75
4.7	Data set details page – relationship sections: (1) data set links, (2) data set collections and (3) data set recommendations. Shows (a) a data set search box . . . . .	75
4.8	Full search interface component for finding data sets . . . . .	77
4.9	Quick search interface component for performing searches by data set name, metadata parameter and file name . . . . .	77
4.10	Slices from the x-y and y-z planes extracted from a CT scan of a material using the image browser plug-in . . . . .	77
4.11	MDC reporting example: a material information sheet showing a material’s composition (generated from the data set’s metadata) and its microstructure (using files in the data set) . . . . .	78
4.12	Material information sheet produced using the data set reporting plug-in . . . . .	78
4.13	Tensile test report produced using the data set reporting plug-in . . . .	80
4.14	Fatigue test report produced using the data set reporting plug-in . . . .	80
4.15	Three slices in the x-y, x-z and y-z planes extracted from a CT scan of a toy car using the CT image browser plug-in . . . . .	81
4.16	Performance results showing the effect of using the file monitoring service on a file share . . . . .	84
5.1	Metadata database schema in data management model . . . . .	97
5.2	Data set details page, (1) security section . . . . .	98
5.3	Data set metadata editing (reordering and nesting) . . . . .	99
5.4	Browsing the contents of a tab delimited file . . . . .	100
5.5	Scatter plot comparing two tensile tests from two delimited data files . . . . .	101
5.6	Comparing the compositions of materials using the graphing feature (bar chart) . . . . .	101
5.7	Comparing the compositions of materials using the graphing feature (pie chart) . . . . .	101
5.8	Data set details page, showing (a) data file upload via web site . . . . .	102
5.9	Browsing the contents of a file containing summary data . . . . .	105
5.10	EP2DC with the new HDC model . . . . .	108
6.1	A simplified data life cycle, used as section headings for the case studies . . . . .	117

6.2–6.32	The <i>Introducing Research Data</i> guide . . . . .	118–147
7.1	Data centre evolution, between EP2DC, MDC and HDC projects and future possibilities . . . . .	154
7.2	The data life cycle as illustrated by Humphrey (2006). The stages have been highlighted to indicate which areas have been addressed by the HDC and EP2DC projects; the entire life cycle was discussed in Chapter 6 . . . . .	155
7.3	The knowledge transfer cycle (‘KT Cycle’) in Figure 7.2 (Humphrey 2006) . . . . .	156
B.1	<i>MatDB</i> top-level element . . . . .	161
B.2	<i>Materials</i> type . . . . .	162
B.3	<i>Source</i> type . . . . .	163
B.4	<i>Tests</i> type . . . . .	163
B.5	<i>Uniaxial Creep</i> test data ( <i>Tests</i> type continued) . . . . .	164
B.6	<i>Uniaxial Tensile</i> test data ( <i>Tests</i> type continued) . . . . .	165
B.7	<i>Low Cycle Fatigue Strain</i> test data ( <i>Tests</i> type continued) . . . . .	166
B.8	<i>Specimen</i> element . . . . .	167
B.9	<i>TestCondition</i> element . . . . .	168
B.10	Other elements: (a) <i>Stress</i> , (b) <i>Temperature</i> , (c) <i>Symbol</i> , (d) <i>Reference To Report</i> , (e) <i>Fracture</i> , (f) <i>Orientation Deviation Angle</i> . . . . .	168





# List of tables

---

1.1	Contributions for each chapter . . . . .	7
2.1	Data unit name comparisons, showing the IEC, JEDEC and metric units used to quantify data . . . . .	12
2.2	Text encoding comparison, ASCII (ISO-8859-1) vs EBCDIC Code Page 285-1/697-1 United Kingdom . . . . .	14
2.3	Example metadata entries . . . . .	35
2.4	Relational database tables for example data in Table 2.3 under the conventional database model . . . . .	39
2.5	EAV database tables for example data in Table 2.3 under the EAV database model . . . . .	39
2.6	Some of the structured storage systems . . . . .	41
3.1	List of data centre operations made available via the REST service . .	56
4.1	Descriptions of the metadata database tables . . . . .	67
4.2	Fields in the DatasetParameters table used to describe metadata entries . . . . .	67
4.3	Synchronisation service tests . . . . .	83
4.4	Synchronisation service test statistics . . . . .	83
4.5	Encoding larger tables using the MDC's metadata database . . . . .	92
6.1	Feedback from research data management introduction lectures . . .	148
7.1	Current number of data sets in the production version of the Heterogeneous Data Centre . . . . .	152



# Declaration

---

I, Mark Scott, declare that the thesis entitled *Research data management* and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as:
  - Tim Austin, Mark Scott, Steven Johnston, Philippa Reed and Kenji Takeda. 2011. ‘EP2DC – an EPrints module for linking publications and data’. In *Proceedings of PV 2011: Ensuring Long-Term Preservation and Adding Value to Scientific and Technical Data*. Toulouse, France, 15th November;
  - Mark Scott, Richard P. Boardman, Philippa A. Reed, Tim Austin, Steven J. Johnston, Kenji Takeda and Simon J. Cox. 2014. ‘A framework for user driven data management’. *Information Systems* 42 (June): 36–58. doi:10.1016/j.is.2013.11.004;
  - Mark Scott, Richard P. Boardman, Philippa A. Reed and Simon J. Cox. 2014. ‘Managing heterogeneous datasets’. *Information Systems* 44 (August): 34–53. doi:10.1016/j.is.2014.03.004;
  - Mark Scott, Richard P. Boardman, Philippa A. Reed and Simon J. Cox. 2013b. ‘Research data management education for future curators’. *The International Journal of Digital Curation* 8 (1): 288–94. ISSN: 1746-8256. doi:10.2218/ijdc.v8i1.261;

- Mark Scott, Richard P. Boardman, Philippa A. Reed, Dorothy Byatt and Simon J. Cox. 2013. ‘Research data management education for future curators’. In *8th International Digital Curation Conference*. Amsterdam, Netherlands: The Digital Curation Centre, 14th January;
  - Wendy White, Simon Cox, Graeme Earl, Mark Scott, Dorothy Byatt and Steve Hitchcock. 2013. ‘Working collaboratively with PhD and early career researchers: Agents for change’. In *8th International Digital Curation Conference*. Amsterdam, Netherlands: The Digital Curation Centre, 14th January–17th January.
- other contributions:
    - Simon J. Cox, James T. Cox, Richard P. Boardman, Steven J. Johnston, Mark Scott and Neil S. O’Brien. 2014. ‘Iridis-pi: A low-cost, compact demonstration cluster’. *Cluster Computing* 17, no. 2 (22nd June): 349–58. ISSN: 1386-7857. doi:10.1007/s10586-013-0282-7;
    - Dorothy Byatt, Mark Scott, Gareth Beale, Simon J. Cox and Wendy White. 2013. ‘Developing researcher skills in research data management: Training for the future – A DataPool project report’. Project Report. Accessed 26th May 2014. <http://eprints.soton.ac.uk/351026/>;
    - S. J. Cox, R. P. Boardman, L. Chen, M. Duta, H. Eres, M. J. Fairman, Z. Jiao et al. 2002. ‘Grid services in action: Grid enabled optimisation and design search’. In *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing: HPDC-11*. Edinburgh, Scotland, 24th July–26th July. doi:10.1109/HPDC.2002.1029943.

Signed: \_\_\_\_\_ Date: \_\_\_\_\_

# Acknowledgements

---

The user requirements report and questionnaire in Section 2.5.1 were prepared by Charlotte Rimer and Philippa Reed. I have quoted from the report and discussed findings to establish requirements.

The development team for the EP2DC project discussed in Chapter 3 also consisted of Sebastien François, Steven Johnston and Tim Austin. I worked on the data storage layer and jointly created the service layer with Steven Johnston.

The work in Chapter 4 and Chapter 5 was fully implemented by me. Its architecture and the image viewer component was a joint collaboration with Richard Boardman, with architecture and design review provided by Simon Cox, and user representation, guidance and oversight provided by Philippa Reed.

The guide in Chapter 6 was written and formatted by me, with supervision from Richard Boardman, Philippa Reed and Simon Cox. A later version was revised with help from Dorothy Byatt in the University library. The case studies were produced with help from Andy Collins, Thomas Mbuya, Kath Soady, Gregory Jasion, Simon Coles and Graeme Earl.

Thanks to Kath Soady, Thomas Mbuya, Rong Jiang, Eleanor Hawkins and Tom Austin for feedback and prototype testing as well as provision of materials engineering data. Thanks to Rob Thompson for supplying the material data sheets. Thanks to Andy Collins and Jane Gibson for their assistance in understanding human genetic medical data and for provision of sample data.

I gratefully acknowledge the following funding and support: the EP2DC and MDC projects in Chapters 3 and 4 were funded by JISC; Chapter 5 was funded by the University of Southampton and ongoing support and funding from Microsoft Research Connections is also gratefully acknowledged; the new educational material presented in Chapter 6 was made possible with Roberts funding and the *Introducing Research Guide* also acknowledges ongoing support from the University of Southampton, The DataPool project, Microsoft, EPSRC, BBSRC, JISC, AHRC and MRC.

My supervisors for this thesis were Simon Cox, Philippa Reed and Richard Boardman. Each provided help and support in their own unique ways and this journey could not have been completed without them. Many thanks also to Kenji Takeda and Tim Austin who started off as supervisors but then ended up on a different path in life.

Invaluable support came from Marcela Andres, Paul Bosson, Andy Chipperfield, Jim Coates, Elizabeth Hart, Gregory Jasion and Neil O'Brien.



# Chapter 1

## Introduction

---

### 1.1 A historical warning



THE BRITISH BROADCASTING CORPORATION (BBC) began transmitting on 2 November 1936 from London and became the world's first regular public, high-definition, television service (Pawley 1961). In the beginning, much of the content was only ever broadcast once and was never retained. Recording it to film was difficult; later, when videotape was available its cost was prohibitive.

Television was only considered secondary to other art-forms such as the radio or the theatre (Pringle 1971, 65). Pringle (1971) suggests that some could see its value, but still took an elitist view, arguing that only critics with the correct training and experience were in a position to decide what was worthy.

Art is not a democracy run on the proportional representation principle. No referendum will decide the worth of a work. Only time will do that. Posterity is the ultimate judge. Meanwhile the critic makes his assessment of what posterity will decide, and he makes it on the basis of training and experience. (Worsley 1970)

In 1963, the BBC broadcast its first episode of the *Doctor Who* television serial. Many of the tapes were not retained and more than a hundred episodes were lost. At the time, the BBC could not possibly know how popular and valuable these episodes might later become, and that the television series would still be popular fifty years later.

Nowadays, the BBC collects all material it produces in its archives and now has over 600 000 hours of television material stored in its purpose built premises at Perivale, London, and is engaged in a search to find missing material. The media recently reported that nine missing episodes of *Doctor Who* had been uncovered in Nigeria to much celebration; they are still missing 97. (Masters 2013)

In scientific research, we could learn a lot from this example. A materials engineer returning from an offsite facility with terabytes of data, often finds themselves in the same position as the BBC did in the early days of television, with too much data and nowhere to store it, repeating the same mistakes. As Worsley states about art, posterity is the ultimate judge of something's worth. It is only with hindsight that we discover its full value.



## 1.2 ‘Standing on the shoulders of giants’

Every discipline, from the arts and humanities to physics, is increasingly using computing data to drive forward its goals. Medicine might use it for recording the statistics of a particular drugs trial; physicists have complex experiments – such as the Large Hadron Collider at CERN – producing massive quantities of data on an hourly basis; and archaeologists meticulously preserve digital records of excavation sites and artefacts.

Hilbert and López (2011) estimated that the worldwide capacity for storing digital information in 2007 was 276 exabytes – equivalent to 276 billion billion ( $10^{18}$ ) bytes. A similar estimate from private sector research company International Data Corporation (IDC), sponsored by EMC Corporation, put the figure at 264 exabytes and calculated that all the data created and replicated in the ‘digital universe’ was 281 exabytes (Gantz et al. 2008) and claimed that, since 2007, more data is produced than can actually be stored. Much of this data is transient – some data is deleted when it is no longer required, some might be transformed into other formats, and some might be processed and the raw data discarded.

To put this into perspective, 281 exabytes, distributed over the 2.7 billion people reported to be using the internet in 2013 by ITU Telecommunication Development Bureau (2013), would mean that each user would have created an average of 104 gigabytes of data, the equivalent of over three million books each.<sup>1</sup>

By 2010, it was estimated that the amount of data created and replicated had passed one zettabyte – a thousand exabytes or a billion trillion ( $10^{21}$ ) bytes – and that 1.8 zettabytes would be created in 2011, projecting 35 zettabytes by 2020 (J. F. Gantz and Reinsel 2011; Fox and Harris 2013). J. Gantz and Reinsel (2012) estimated that 2.8 zettabytes of data would be created in 2012 and forecast that this would reach 40 zettabytes by 2020.

To store 2.8 zettabytes of data would require 1400 million mid-priced consumer-grade portable hard disks.<sup>2</sup> More details on the terminology of bytes, gigabytes, exabytes and zettabytes will be given in Section 2.3.

For centuries research and knowledge has been shared through the publication and dissemination of books, papers and scholarly communications but, moving forward, much of our understanding relies on (large scale) data sets which have been collected or generated as part of this scientific process of discovery. The ‘dwarfs [*sic*] on the shoulders of giants’ metaphor (John of Salisbury 1955, p. 167; Sandys 1906, p. 531), most famously used by Isaac Newton<sup>3</sup>, refers to how we rely on the achievements of those who preceded us. With the volume of scientific research data increasing rapidly (Hey and Trefethen 2003; Johnston, Fangohr and Cox 2008), and institutions and funding councils spending large amounts of money on the resources required to produce it, it is critical that we ensure that, once collected or generated, future generations can benefit from the data we produce.

---

1. 3 328 000 books, assuming an average of 5 characters per word, 250 words of text per page, 250 pages per book and one character equals one byte.

2. Mid-priced example: £94.50 incl. VAT for ‘2 TB Western Digital My Book USB 3.0/2.0 For Mac’ (<http://www.scan.co.uk> on 03/04/2013).

3. ‘If I have seen further it is by standing on ye sholders [*sic*] of giants’ (Newton 1959, p. 416; Merton 1965, p. 31).

## 1.3 Data sharing

The researchers of current generations can also benefit from this data. Their data can be expensive or time consuming to produce, creating a dilemma: whether to share it with others and lose a competitive advantage or keep it private, limiting the contributions that others could bring. The consensus is to hold the data for a period and then release to everyone. This is in the interest of funding bodies who may not be willing to finance the same experiments more than once, but while some publishers of journals do stipulate that accompanying data such as microarray data must also be provided with a paper (Nature 2013), this is not yet commonplace.

There have been some high-profile examples of where data sharing would be in the public interest. A publication in *The Lancet* (Wakefield et al. 1998) falsely linked bowel disease and autism to the measles, mumps and rubella vaccine and triggered public panic reducing the uptake of the valuable medicine by patients. In a second example, the accuracy of the CRUTEM3 data set was brought into question when emails from Professor Phil Jones were stolen from The Climate Research Unit at the University of East Anglia's computers which allegedly showed that he was part of a conspiracy to hide evidence that did not fit his views. The incident was investigated by the House of Commons Science and Technology Committee (House of Commons Science and Technology Committee 2010) which concluded Professor Jones acted in line with common practice and cleared him of any wrong-doing, but recommended that climate scientists should take steps to make raw data available or to indicate where data has been used that is not published, in order to make 'the quality and transparency' of their science 'irreproachable'.

Sharing data sets among colleagues within a research team is not always easy as portable hard drives are often used to store the data which presents a problem with the sharing and capture of data, particularly with the data being offline or while the data is still in use by its owner. Storing and retaining the data at all is better than removing old data sets to make way for new data.

## 1.4 Research data management

Research data management is concerned with looking after the data throughout its entire lifecycle. Figure 1.1 shows one way of representing this data lifecycle (Higgins 2008) produced by the Digital Curation Centre (DCC) who provide advice, guides, support and training about research data management to UK higher education institutes and also publish the *International Journal of Digital Curation* relating to work concerning research data management. Other ways of looking at this lifecycle are presented in Chapter 6.

Some research councils have begun including mandatory data retention policies (Jones 2012). To show compliance with these, data management plans (DMPs) are required to be created to document what data will be created and how, and detail how it will be shared and preserved. The DCC have developed the DMPonline tool to assist with the preparation of these. It focusses on collecting information relevant to the requirements of UK funders with

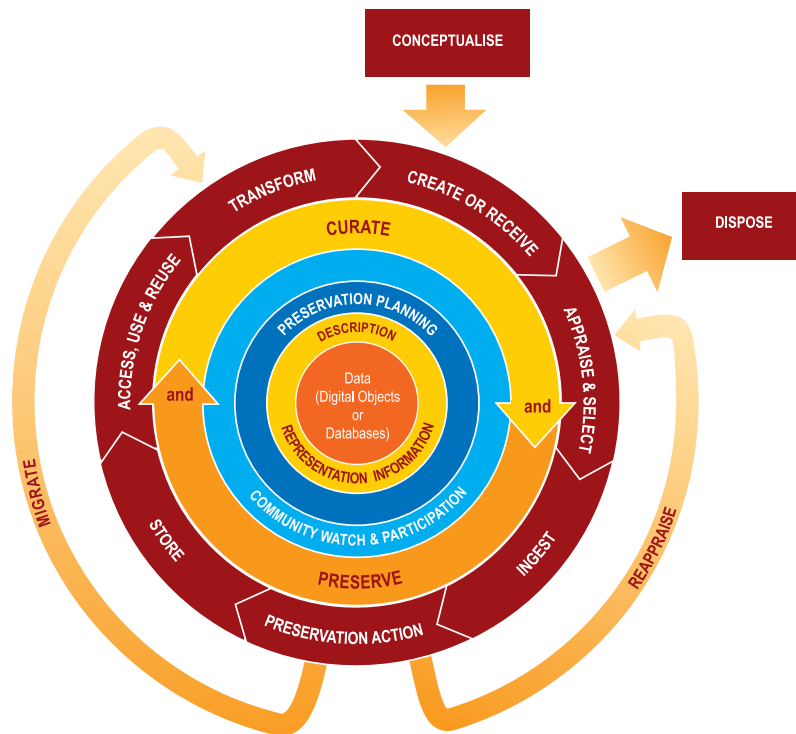


Figure 1.1: The Digital Curation Centre's data lifecycle (Higgins 2008).

some provision for overseas funding bodies. Another tool called the DMPTool was developed for US institutions with the help of the DCC. (Sallans and Donnelly 2012)

One critical stage of this lifecycle that was alluded to in Section 1.1 is data preservation which will be discussed in Section 2.3.1. The Digital Preservation Coalition (DPC) aim to raise awareness of the issues surrounding this stage and help its members to achieve their preservation goals through knowledge exchange and capacity building (Beagrie 2000). Educating researchers about these concerns is discussed to Chapter 6.

There has been a lot of effort put into what happens to data on its publication and how to preserve it in the long term. The International DOI Foundation funded the introduction of Digital Object Identifiers (DOIs) which are used to identify and track objects on the internet. They are commonly used for tracking published papers but are being used for data sets too; this is being supported with the efforts of DataCite. DataCite enables organisations to register research data sets and assign persistent identifiers (Brase 2009). Other efforts on data publication such as the DataDryad repository will be discussed in Section 2.9.1; many of these publication repositories assign DOIs to their data sets. Chapter 3 will look at how a document repository can be linked to a data set stored in a data repository but Chapters 4 and 5 will explore tools for looking after data before publication.

## 1.5 Data capture

When work on a paper is completed it has become common practice to store a copy of it within a document repository in order for others to share and benefit, but in a white paper produced by the DRIVER project it was reported that 90% of the records held in the digital

repositories of the institutions they surveyed related to articles, books, theses, proceedings and working papers (Eijndhoven and Graaf 2007). Non-textual material (images, videos, primary data sets and music) only accounted for 5% of records, with the remaining 5% of records categorised as ‘other materials’. These statistics suggest that any data used to produce the research is not always stored, making it more difficult for colleagues within an institution, or in the wider community, to fully comprehend or progress the work.

Chapter 3 discusses how data in a data repository can be linked to a publication, and how data can be uploaded and validated when depositing a paper.

## 1.6 Materials engineering data

Materials engineers produce diverse data when testing materials and investigating material failure, creating everything from text data to microfocus computed tomography data files in their day-to-day research, with a lot of these data sets having important relations to each other. Use of portable hard disks is common – for example, during visits to off-site synchrotron facilities or use of on-site equipment – and many of these data sets are not being shared which is sometimes attributable to confidentiality concerns, but often is due to a lack of resources or motivation (Rimer and Reed 2009, 59–63).

Some engineers need to store very structured data such as MatDB (Ojala and Over 2008), a standard schema able to represent data from tensile, creep and fatigue tests in XML. Others just want a large data store where they can save raw, unprocessed image data until they are ready to use it. A few require the ability to create information sheets about the materials they use and link them to the results of experiments, and some just want to store raw data CSV files containing their results in order to support findings in a paper.

The need for tools which support materials engineering is underlined by the United States Government’s *Materials Genome Initiative for Global Competitiveness* which ‘aims to reduce development time by providing the infrastructure and training that American innovators need to discover, develop, manufacture, and deploy advanced materials in a more expeditious and economical way’ (National Science and Technology Council 2011). Tinkle et al. (2013) discussed five ways in which materials scientists can improve their data sharing, specifically referring to the Materials Genome Initiative:

1. Learn from other initiatives
2. Incentivise sharing
3. Embrace uncertainty
4. Make simulations reproducible
5. Probe the infinite variety

The relevant ones for us in this thesis are finding ways to attract users to deposit their data (‘Incentivise sharing’) which we will discuss in Chapter 5, and ‘Probe the infinite variety’ – referring to the infinite variety of materials – we will present a model in Chapter 4 whose aim

is to allow engineers to store the data collected from tests, whatever the material. Chapter 2 discusses other data management approaches (addressing ‘Learn from other initiatives’).

## **1.7 Thesis outline**

The variety of materials engineering data set types, sizes, metadata and relationships between data sets create a complex problem in terms of storing, managing and retrieving. In the wider research community, data capture and curation from birth to publication is difficult to achieve or is not done.

Providing researchers with the correct tools to look after their data and educating them on the value of data is becoming much more important but managing data sets within a research environment at a University whilst the data is still in use presents some challenges that are not addressed by a simple document repository. Many systems provide excellent data management features suitable for the final publication of data, but users can be daunted by the deposit processes particularly whilst still working with their data (Rimer and Reed 2009, 55–56), and capture of data produced by equipment such as X-ray computed tomography scanners and scanning electron microscopes would also require manual interaction with the repository to supply any mandatory metadata.

This thesis discusses the capture, sharing and curation of data from a number of perspectives. The first is to show how data might be shared when uploading a research publication to an institutional document repository by uploading the data to an engineering data repository at the same time and linking the data to the publication. This work was part of the EP2DC project which led to the Materials Data Centre project (MDC) and the expansion of the supporting data repository’s features and the development of a data management model permitting storage of a wider range of materials engineering data, accompanying metadata and relationships. The heterogeneous nature of the data management approach shows there is potential for managing data from other disciplines; to show this, we will also look at data from human genetics and expand the data repository further with the Heterogeneous Data Centre (HDC). Finally, we present a number of case studies and a guide which were developed during the course of the project to educate researchers on the value of their data.

### **1.7.1 Main research questions and thesis objectives**

In this thesis, we will research the following key questions:

1. Without disrupting users’ existing workflows, what is required to capture the prepublication data and metadata of materials engineers whilst it is still in use?
2. Given the heterogeneity of materials engineering data, can this framework cope with future needs and be adapted for use in different disciplines?
3. How would data in such a system be linked to a research publication at a later date?
4. Is it necessary to educate users about data management issues and how would this be done?

**Table 1.1:** Contributions for each chapter.

Chapter	Contribution	
Chapter 3	Deposit of validated materials data in a data repository from a document repository, maintaining the relationship and displaying recommendations to other deposited data sets	<sup>4</sup>
Chapter 4	Use of compoundable metadata database to annotate pre-publication, in-use, materials engineering data sets held in a monitored file store	<sup>5</sup>
Chapter 5	Plug-ins allowing materials engineers to work with data files across data sets; automatic metadata ingestion; a complete system that allows in-use, pre-publication heterogeneous data to be stored	<sup>6</sup>
Chapter 6	Research data management education material	<sup>7, 8</sup>

To address these questions, we establish the following objectives for this thesis:

1. To develop a platform-independent framework for the management of heterogeneous data and metadata.
2. To implement this framework with off-the-shelf technology, to build an extensible system that naturally fits into the existing workflows of users, to test the first key question.
3. To develop the framework's implementation with real data and feedback from users from materials engineering.
4. To explore tools that allow users to query, report on and work with deposited data.
5. To test the framework's implementation with other types of data, to confirm the key question of whether the framework can cope with future needs or be adapted for use in other disciplines.
6. To integrate the system with established document publishing technologies, to demonstrate data management throughout its lifecycle and to test the question of whether such a system precludes the ability to publish data with a research publication at a later date.
7. To develop educational material to encourage users to begin considering the issues of data management, to test the final key question about user awareness of data management issues.

### 1.7.2 Structure

The thesis is structured in the following way. Supporting publications are given as footnotes. Table 1.1 summarises the contributions made in each chapter.

**Chapter 2, 'Computing data and associated technologies':** This chapter discusses what data is and how it is traditionally managed. Some examples of materials engineering data are given which provided the guiding requirements for the design of the data management framework presented in this thesis. Existing projects that have managed either heterogeneous data or large amounts of data, and the approaches taken by them, are investigated. The chapter discusses why another data management system is required.

**Chapter 3, ‘Data publishing with EPrints’:** This chapter describes the EP2DC project and how data can be linked to a publication. The research explores how a document repository can be extended to accept data upload and transmit it to a discipline specific data repository, and how that data can be linked to a publication with the EP2DC system.<sup>4</sup>

**Chapter 4, ‘A model for managing materials data’:** This chapter describes in detail the architecture of the model and some its design features, and also shows the framework in use with materials data. The chapter shows how the data repository in the EP2DC system, as part of the Materials Data Centre (MDC) framework, can be improved to provide the following features:<sup>5</sup>

- Storage and sharing of any data files, limited in size only by the file system.
- The addition of metadata to a data set, with the ability for users to define their own data structures, with nesting.
- Support of the relationships between data sets.
- Allow for predefined metadata to be created as a template and for metadata to be copied between data sets.
- Encourage use of the system by providing data set recommendations, blogs, wikis and message boards.
- Allow plug-ins, in order for the system to provide customised reports and tools depending on the data.
- Support of the following data types: Tensile test, Fatigue test, Fractography and Microfocus computed tomography data.

**Chapter 5, ‘A model for managing heterogeneous data’:** This chapter expands the system further to support heterogeneous data and add features to encourage user uptake. The framework is shown in use with human genetics data. Features concentrate on four areas:<sup>6</sup>

- Straightforward data deposition process.
- Metadata and metadata tools.
- Data set discoverability.
- Data set tools adding value to users, to encourage uptake and induce users to provide data and metadata.

**Chapter 6, ‘Research data management education’:** Building data management systems cannot guarantee users will change the way they work. Teaching researchers the importance of their scientific data is another part of this work and this chapter discusses a number of case studies and a guide that was prepared to help researchers think about their data, particularly early in their careers.<sup>7, 8, 9</sup>

**Chapter 7, ‘Summary and outlook’:** Finally, this chapter discusses the evolution of the work presented in this thesis and whether the objectives were met, summarises the findings and considers how the data management model could be taken further.

The next chapter will introduce what data is and how it is traditionally managed.

- 
4. Tim Austin, Mark Scott, Steven Johnston, Philippa Reed and Kenji Takeda. 2011. ‘EP2DC – an EPrints module for linking publications and data’. In *Proceedings of PV 2011: Ensuring Long-Term Preservation and Adding Value to Scientific and Technical Data*. Toulouse, France, 15th November.
  5. Mark Scott, Richard P. Boardman, Philippa A. Reed, Tim Austin, Steven J. Johnston, Kenji Takeda and Simon J. Cox. 2014. ‘A framework for user driven data management’. *Information Systems* 42 (June): 36–58. doi:10.1016/j.is.2013.11.004.
  6. Mark Scott, Richard P. Boardman, Philippa A. Reed and Simon J. Cox. 2014. ‘Managing heterogeneous datasets’. *Information Systems* 44 (August): 34–53. doi:10.1016/j.is.2014.03.004.
  7. Mark Scott, Richard P. Boardman, Philippa A. Reed and Simon J. Cox. 2013b. ‘Research data management education for future curators’. *The International Journal of Digital Curation* 8 (1): 288–94. issn: 1746-8256. doi:10.2218/ijdc.v8i1.261.
  8. Mark Scott, Richard P. Boardman, Philippa A. Reed, Dorothy Byatt and Simon J. Cox. 2013. ‘Research data management education for future curators’. In *8th International Digital Curation Conference*. Amsterdam, Netherlands: The Digital Curation Centre, 14th January.
  9. Wendy White, Simon Cox, Graeme Earl, Mark Scott, Dorothy Byatt and Steve Hitchcock. 2013. ‘Working collaboratively with PhD and early career researchers: Agents for change’. In *8th International Digital Curation Conference*. Amsterdam, Netherlands: The Digital Curation Centre, 14th January–17th January.





## Chapter 2

# Computing data and associated technologies

---

### 2.1 Introduction



THE FINDINGS OF THIS THESIS are based on the work done as part of the Materials Data Centre (MDC) project and its associated project EP2DC. The MDC (Reed, Austin and Takeda 2009; Scott, Boardman, Reed, Austin et al. 2014) was a UK Government (JISC) funded project to establish a repository promoting data capture and management in the engineering materials domain (Rimer and Reed 2009, 5), while EP2DC (Reed et al. 2009; Austin et al. 2011) concentrated on the submission of a paper to a document repository along with its accompanying data to a data repository and maintaining the relationship between them.

Before discussing research data specific to the materials engineering discipline, this chapter will discuss data in general and some of the hardware and software that can be used to store it, showing that there is a wide variety of types of data which require careful management for long-term preservation and interoperability. Section 2.5 and Section 2.6 discusses the questionnaires, user requirements document, interviews and types of engineering data that were used to establish requirements for the Materials Data Centre project. We will then discuss data management technologies and projects.

### 2.2 Terminology

*Data* is defined as ‘Quantities, characters, or symbols on which operations are performed by a computer, considered collectively’ (Oxford English Dictionary 2012). Data can be the structures making up a file on disk but also the quantities, characters and other concepts that the structures represent to a software application or user. Data can be used as a mass noun, in the same way as the related terms *information* and *knowledge*, and therefore has no form of plural. Data is also the plural of *datum* so, when using the term in this way, it should be treated as plural.

*Information* is said to come from the processing of the data. In general, data can be considered as raw numbers whereas information is those numbers after they have been processed. Perhaps it is a set of measurements of the length of earthworms that have been grouped by

**Table 2.1:** Data unit name comparisons, showing the IEC, JEDEC and metric units used to quantify data.

(a) Base two units, used to quantify memory sizes.					(b) Base ten units, used to quantify hard disk drive sizes.				
IEC unit	JEDEC unit		Number of bytes		$2^x$	Metric unit	Number of bytes		$10^x$
byte	B	byte	B	1	$2^0$	byte	B	1	$10^0$
kibibyte	KiB	kilobyte	KB	1024	$2^{10}$	kilobyte	KB	1000	$10^3$
mebibyte	MiB	megabyte	MB	1 048 576	$2^{20}$	megabyte	MB	1 000 000	$10^6$
gibibyte	GiB	gigabyte	GB	1 073 741 824	$2^{30}$	gigabyte	GB	1 000 000 000	$10^9$
tebibyte	TiB			1 099 511 627 776	$2^{40}$	terabyte	TB	1 000 000 000 000	$10^{12}$
pebibyte	PiB			1 125 899 906 842 624	$2^{50}$	petabyte	PB	1 000 000 000 000 000	$10^{15}$
exbibyte	EiB			1 152 921 504 606 846 976	$2^{60}$	exabyte	EB	1 000 000 000 000 000 000	$10^{18}$
zebibyte	ZiB			1 180 591 620 717 411 303 424	$2^{70}$	zettabyte	ZB	1 000 000 000 000 000 000 000	$10^{21}$
yobibyte	YiB			1 208 925 819 614 629 174 706 176	$2^{80}$	yottabyte	YB	1 000 000 000 000 000 000 000 000	$10^{24}$

country and compared. The measurements make up the data, but the grouped and compared data is information.

*Knowledge* comes from the interpretation of a set of information. Information, by contrast is processed data and facts, without the experience or understanding to comprehend it. In computer science, knowledge can be a set of rules that help software make decisions.

Grasping the differences between these terms can be difficult because they often hold different meanings to different people. For example, a software developer's concept of data might be different to the software's user. The developer's view of data when saving a user's document is to store enough particulars to permit the document to be reopened. The developer will interpret the data they have stored to extract information in order to reconstruct the document. To the user, the data is what has been entered into the document and the information is what can be derived from manipulation or studying of the data. At an even lower level, the hardware that stores the file only concerns itself with a series of binary digits (discussed in Section 2.3) when saving and retrieving the file.

We consider data to be the files that researchers produce as part of their work, as well as the metadata that is used to annotate that data. We also look at the relationships between pieces of data.

## 2.3 Computing data

Computers use binary for representing data. Binary is a system of numerical notation with a base of two which means it uses only zeros and ones. A single zero or one is known as a bit, and groups of bits are known as bytes, where a byte is usually eight bits.

Prefixes for multiples of bytes include kilo- for 1000 ( $10^3$ ), mega- for 1 000 000 ( $10^6$ ) and giga- for 1 000 000 000 ( $10^9$ ). Due to the use of binary, which is base two and not base ten, these multiples are often not the most convenient, so other prefixes exist including kibi- for 1024 ( $2^{10}$ ), mebi- for 1 048 576 ( $2^{20}$ ) and gibi- for 1 073 741 824 ( $2^{30}$ ). Confusingly, the kilo, mega and giga group of prefixes are frequently used in the place of the kibi, mebi and gibi prefixes.

Table 2.1 summarises the various unit names and the number of bytes each represents. In this thesis, the base two IEC units (KiB, MiB, GiB) in Table 2.1(a) or the base ten metric units (KB, MB, GB) in Table 2.1(b) are used when specifying data sizes.

### 2.3.1 File data

File data is encoded using binary sequences understood by the software application creating the file allowing it to represent text data, formatting, image data or any other required information. So, for example, a spreadsheet application must save enough data to recreate a document without any loss of calculations, numbers, functions, formatting or any other features used by the creator.

The significance of file formats can be demonstrated by looking at the storage of text data. When only text data is required, a specialist text format is often used which is understood by many of the tools provided with the computer's operating system. However, there are a number of standards for encoding text files. In the early days of computing, EBCDIC (IBM Corporation 1995) was popular on IBM mainframes but this was gradually supplanted by ASCII (American National Standards Institute 1986). This, however, had limitations, especially with foreign alphabets so a number of alternative encodings and extensions to ASCII were created to cope with extra characters, such as HZ, GBK and BIG5 encoding formats, and Unicode.

Examples of text files are given below:

**CSV:** A file containing a list of data items with commas delimiting the fields. Spreadsheets can be represented in this format.

**XML:** A structured data file represented in a format that a computer can understand, while remaining human readable (Bray et al. 2008), which follows the structure defined in an XML schema (Fallside and Walmsley 2004). An XML schema is defined with elements, types and attributes. Types define the data used in the schema and can be simple, such as an integer, or complex, such as a combination of other elements and attributes.

**HTML:** A structured data file, with a format similar to XML, used to program the layout of web pages.

**Source code:** To create software, a text file must first be created containing instructions written in a language that can be translated by a compiler.

**Data file:** Text data produced by software or appliance in a structured format it can understand, but still readable by a human with minimal effort or tools.

**Other text file:** Other examples of text data include notes written by a computer user, an email and a log file produced by computer software.

**Table 2.2:** Text encoding comparison, ASCII (ISO-8859-1) vs EBCDIC Code Page 285-1/697-1 United Kingdom.

EBCD ASCII			EBCD ASCII			EBCD ASCII			EBCD ASCII			EBCD ASCII		
0	NUL	NUL	52	PP	4	104	Ç	h	156	æ	ST	208	}	Ð
1	SOH	SOH	53	TRN	5	105	Ñ	i	157	,	OSC	209	J	Ñ
2	STX	STX	54	NBS	6	106		j	158	Æ	PM	210	K	Ò
3	ETX	ETX	55	EOT	7	107	,	k	159	□	APC	211	L	Ó
4	SEL	EOT	56	SBS	8	108	%	l	160	μ	NBSP	212	M	Ô
5	HT	ENQ	57	IT	9	109	_	m	161	˘		213	N	Õ
6	RNL	ACK	58	RFF	:	110	>	n	162	s	¢	214	O	Ö
7	DEL	BEL	59	CU3	;	111	?	o	163	t	£	215	P	×
8	GE	BS	60	DC4	<	112	ø	p	164	u	□	216	Q	Ø
9	SPS	HT	61	NAK	=	113	É	q	165	v	¥	217	R	Ù
10	RPT	NL	62		>	114	Ê	r	166	w		218	¹	Ú
11	VT	VT	63	SUB	?	115	Ë	s	167	x	§	219	ú	Û
12	FF	NP	64	SP	@	116	È	t	168	y	¨	220	ü	Ü
13	CR	CR	65	RSP	A	117	Í	u	169	z	©	221	ù	Ý
14	SO	SO	66	â	B	118	Î	v	170	i	ª	222	ú	Þ
15	SI	SI	67	ä	C	119	Ï	w	171	ç	«	223	ÿ	ß
16	DLE	DLE	68	à	D	120	Ì	x	172	Ð	¬	224	\	à
17	DC1	DC1	69	á	E	121	‘	y	173	Ý	SHY	225	÷	á
18	DC2	DC2	70	â	F	122	:	z	174	Þ	®	226	S	â
19	DC3	DC3	71	ã	G	123	#	{	175	®	˘	227	T	ã
20	RES ENP	DC4	72	ç	H	124	@		176	¢	°	228	U	ä
21	NL	NAK	73	ñ	I	125	,	}	177	[	±	229	V	å
22	BS	SYN	74	\$	J	126	=	˘	178	¥	²	230	W	æ
23	POC	ETB	75	.	K	127	"	DEL	179	.	³	231	X	ç
24	CAN	CAN	76	<	L	128	Ø	PAD	180	©	´	232	Y	è
25	EM	EM	77	(	M	129	a	HOP	181	§	μ	233	Z	é
26	UBS	SUB	78	+	N	130	b	BPH	182	¶	¶	234	²	ê
27	CU1	ESC	79		O	131	c	NBH	183	$\frac{1}{4}$	·	235	Ô	ë
28	IFS	FS	80	&	P	132	d	IND	184	$\frac{1}{2}$	¸	236	Ö	ì
29	IGS	GS	81	é	Q	133	e	NEL	185	$\frac{3}{4}$	²	237	Ò	í
30	IRS	RS	82	ê	R	134	f	SSA	186	^	°	238	Ó	î
31	IUS ITB	US	83	ë	S	135	g	ESA	187	]	»	239	Õ	ï
32	DS	SP	84	è	T	136	h	HTS	188	˘	$\frac{1}{4}$	240	0	ð
33	SOS	!	85	í	U	137	i	HTJ	189	¨	$\frac{1}{2}$	241	1	ñ
34	FS	"	86	î	V	138	«	VTs	190	´	$\frac{3}{4}$	242	2	ò
35	WUS	#	87	ï	W	139	»	PLD	191	×	ç	243	3	ó
36	BYP	\$	88	ì	X	140	ð	PLU	192	{	À	244	4	ô
37	LF	%	89	ß	Y	141	ý	RI	193	A	Á	245	5	õ
38	ETB	&	90	!	Z	142	þ	SS2	194	B	Â	246	6	ö
39	ESC	,	91	£	[	143	±	SS3	195	C	Ã	247	7	÷
40	SA	(	92	*	\	144	°	DCS	196	D	Ä	248	8	ø
41	SFE	)	93	)	]	145	j	PU1	197	E	Å	249	9	ù
42	SM	*	94	;	^	146	k	PU2	198	F	Æ	250	³	ú
43	CSP	+	95	¬	_	147	l	STS	199	G	Ç	251	Û	û
44	MFA	,	96	-	‘	148	m	CCH	200	H	È	252	Ü	ü
45	ENQ	-	97	/	a	149	n	MW	201	I	É	253	Ù	ý
46	ACK	.	98	Â	b	150	o	SPA	202	SHY		254	Ú	þ
47	BEL	/	99	Ä	c	151	p	EPA	203	ô	Ë	255	EO	ÿ
48		0	100	À	d	152	q	SOS	204	ö	Ì			
49		1	101	Á	e	153	r	SGCI	205	ò	Í			
50	SYN	2	102	Ã	f	154	a	SCI	206	ó	Î			
51	IR	3	103	Å	g	155	o	CSI	207	õ	Ï			

### Extended Binary Coded Decimal Interchange Code (EBCDIC)

EBCDIC was an early form of encoding used on IBM mainframes. It uses eight bits to encode text which gives it the ability to encode 256 characters, but some of these are used as control characters such as a line feed and the tab character. EBCDIC has many variants, known as code pages, which gives it some international support. The codepage for use in the UK is shown in Table 2.2 alongside ASCII.

### American Standard Code for Information Interchange (ASCII)

ASCII originally used seven bits to encode text, giving it 128 characters including control characters. A number of national variants were defined, replacing some characters with regional specific symbols such as \$ being replaced with ¥ in China and ¤ in Sweden, and in the UK, # was replaced with £.

In order to encode larger alphabets such as Chinese, multiple bytes were used to encode a single character. Hanzi (HZ) (Lee 1995) encoding allowed the GB2312 Chinese character set (containing 6763 Chinese characters) to be encoded using two bytes by surrounding the Chinese characters with ‘~{’ and ‘~}’. Anything outside these is interpreted as ASCII, but bytes inside the braces are interpreted as Chinese characters two bytes at a time. Another common Chinese encoding system was Extended Unix Code (EUC).

Several different eight bit versions of ASCII evolved, using the additional 128 characters for international support, punctuation marks or even graphical symbols in the case of IBM's code page 437. Encoding the Chinese alphabet was achieved using standards such as GBK, GB18030, and BIG5. GBK used two bytes for encoding the Chinese characters where the first byte had a value of 129 or higher. If the value was less than 128 it was treated as 7-bit ASCII. GBK extended the GB2312 character set, adding new characters as well as supporting traditional Chinese characters. After GBK came GB18030, adding further characters and using one, two or four bytes depending on the character being encoded.

### The Unicode Standard

The most current text encoding standard is known as The Unicode Standard (The Unicode Consortium 2012) which defines a number of text encoding methods as well as the character sets that can be encoded. Unicode defines many characters, including foreign alphabets, so it can support Chinese, Arabic and other languages. Version 6.2 of the standard defines 110 117 characters in total but has space for 1 114 112. The first 256 characters defined by The Unicode Standard have been kept the same as ASCII (specifically the ISO-8859-1 variant). There are a number of different ways of encoding these characters, such as the Unicode Transformation Formats (UTF) including UTF-8 and UTF-16, and the 2-byte Unicode Character Set (UCS-2). UTF-8 is the closest to ASCII as it encodes the first 127 characters of the character set using only one byte in the same way as seven bit ASCII. Higher character values require encoding using multiple bytes.

## Interoperability

To highlight the problems with differences in text encoding formats, we can look at the differences between ASCII and EBCDIC. Table 2.2 shows the EBCDIC and ASCII (ISO-8859-1) character sets side by side. There were a number of main differences between ASCII and EBCDIC, including:

- Different binary encodings for each character, e.g. a byte value of 90 represents ! in EBCDIC and Z in ASCII.
- ASCII was originally seven bits while EBCDIC used eight bits – EBCDIC could represent a larger number of characters.
- EBCDIC and ASCII have lower case, upper case and numbers in a different order.
- In ASCII the upper case alphabet appears consecutively between entries 65 and 90, but in EBCDIC it is between 193 and 233, and interspersed with other characters. This also happens in the lower case alphabet.

These and other differences prevent ASCII and EBCDIC formats from being interchangeable. For example, if the text ‘hippopotamus’ was encoded using ASCII but read back as EBCDIC, the contents would be interpreted as ‘çÑøø?ø?È/\_ÍË’.

To give a slightly more subtle example of incorrect interpretation of text data, Table 2.2 shows ASCII encoded using eight bits and using the ISO-8859-1 standard, but another ASCII standard known as ISO-8859-15 replaces eight characters to permit symbols such as the euro character. Anyone who requires the euro symbol might have saved their text file as ISO-8859-15, but if read back using the ISO-8859-1 character set, € would become ¤.

This highlights the dangers of choosing suitable file formats. ASCII is now gradually being replaced by UTF standards. However, these come with their own problems. A file encoded with ISO-8859-1 is not compatible with UTF-16, which expects characters to be encoded using groups of sixteen bits rather than eight. Unicode can also be encoded with bytes in a different order (known as little endian and big endian), and some formats specify the order by including a special byte sequence known as a byte order mark (BOM) at the start of the file. Tools not expecting this would interpret this sequence as ordinary text.

## Other File Types

Software can encode data using other binary formats that do not follow the standards of EBCDIC, ASCII or UTF, allowing advantages over the text standards such as encryption, better management of space or speed improvements. Files encoded in this way cannot be easily understood by people looking at their contents, except by the program that created them. This means that long term data preservation could be more difficult if the software was no longer available. Searching or indexing them would be difficult without knowledge of their structure and recovering data from a binary file that has been corrupted is more difficult than text files because a text file can often be pieced together and still be valid whereas binary

files can contain critical regions which make the file useless if they are lost. Examples of these types of files include:

**Compressed file:** A file that has repeating data or wasted space removed in order to reduce its size. The file will usually need uncompressing first before it can be used. Text files tend to compress very well, partly because they will have a lot of repetition and they often contain only a limited number of characters from the character set which means fewer bits can be used to represent the same data. Example formats include bzip2, gzip, zip, rar and 7z.

**2D image data:** Examples include png, gif, tiff, jpeg, raw, bmp, svg and cgm.

**Compiled software program:** There is a rich profusion of software programs for many tasks. Software used in materials engineering include ImageJ discussed in Section 2.5.2 and Microsoft Excel discussed in Section 2.6.1.

**Database data file:** Database technologies will be discussed in Section 2.7. A data file is often used by the software for storage.

**Word processor document:** Examples of past and present word processing software include Microsoft Word, AppleWorks, Apple Pages, OpenOffice/LibreOffice/StarOffice, Microsoft Works, Nisus Writer, AmiPro, WordPerfect, WordStar, Protex and FrameMaker. Word Processor documents are typically binary (although some now use XML) containing encoded instructions for font type, size, colour, etc., which means files are often not interchangeable.

## Data Preservation

For any data repository, long-term data preservation must be considered. This section has demonstrated that there is a wide variety of current and obsolete data encoding formats and this can be a concern. For example, the once popular word processing software AmiPro has since been discontinued causing problems for users who used this format in the past. Even opening files between different versions of an application can cause issues if features between versions have changed.

Ensuring critical data is stored in a reliable format is important for data preservation. Using file formats with openly published specifications can provide some protection because the specifications might provide a way to retrieve data from the file if the software is no longer available. Exporting data into a text-encoded format can also help protect vital data but, while this does provide a good way of preserving data, text files can be encoded with different methods (such as ASCII, EBCDIC and Unicode) so it comes with its own problems.

Smit, Hoeven and Giaretta (2011) presents several strategies for digital preservation: using standardised file formats, regular data refreshing (copying of data to new storage media) to protect against data degradation (further discussed in Section 2.4.1), transformation/migration of data from older formats into newer formats, emulation of older software to access



obsolete data formats, and providing metadata about the data to describe its meaning to future generations.

Education of researchers on the aspects of data preservation makes them aware of the issues so they can protect their own data long-term. This is discussed in Chapter 6.

### **2.3.2 Metadata**

Metadata is data about data, such as who created the data and when. It is useful for categorising data to find it later and with the huge amount of data being produced by many domains useful metadata becomes increasingly necessary.

Metadata tends to be standardised in each field. It is used in data files to provide pertinent information about the contents of the data. Microsoft Word documents store the author's name, company and creation/modification date of a document (Microsoft 2006). Portable Document Format (PDF) files (Adobe Systems Incorporated 2008) use two types of metadata: a document information dictionary using a key-value approach and a newer metadata stream stored as XML and complying with the XMP specification (Adobe Systems Incorporated 2005). Some research has been done on extracting metadata from PDF files for ingestion into digital libraries (Marinai 2009). The XMP specification is also used as metadata for image data files, although digital cameras often use the Exchangeable image file format (Exif) (AV&IT Standardization Committee 2010). The usefulness of Exif metadata in digital evidence analysis was shown by Marinai (2009) where extraction of the metadata can show whether an image has been modified.

Archival repositories expect metadata to accompany the item being deposited. Dublin Core (Dublin Core Metadata Initiative 2004) is one of the more commonly used standards but, depending on the discipline, other metadata standards exist. Libraries have historically used MARC (MACHine-Readable Cataloging) (Library of Congress – Network Development and MARC Standards Office 2013a) for encoding metadata, with MARCXML an updated method of transmitting MARC metadata. MARC 21 defines 1908 fields/subfields. However, Moen and Benardino (2003) analysed 419 657 MARC 21 bibliographic records from a sample taken from a WorldCat bibliographic database and found that, out of the available MARC 21 fields/subfields, only 926 were used, and 36 of these accounted for 80% of the total data. MODS (Metadata Object Description Schema) (Library of Congress – Network Development and MARC Standards Office 2013b) is an updated standard which is less complex than MARC but more flexible than Dublin Core which only defines 15 metadata elements. Beall (2005) showed that the quality of metadata in a digital library can sometimes be suspect and discussed a number of places where metadata errors can be introduced.

We will show a flexible metadata model in Section 4.2.2 and discuss the features that help to improve metadata quality in Section 4.7.2. We will show automatic metadata extraction in Section 5.2.6.

## 2.4 Data storage

Storage of computing data requires hardware and software, with criteria such as capacity, speed, reliability and cost to be considered. This section summarises the hardware, local and network file systems and newer technologies such as cloud storage that make data storage possible.

### 2.4.1 Storage hardware

A number of technologies have been developed to support the storage of data, including: (Khurshudov 2001)

**Paper (historical):** e.g. punched card, punched tape

**Magnetic:** e.g. hard disk, floppy disk, tape

**Optical:** e.g. CD, DVD, Blu-ray disk, magneto-optical disk

**Flash memory:** e.g. USB drive, solid state disk, SD card

Each of these has its own advantages and disadvantages, with paper media and the floppy disk being surpassed by newer technologies. The floppy disk was used as portable media but was gradually supplanted by optical media and flash memory USB drives because of higher capacity and performance. (Khurshudov 2001)

#### The hard disk drive

The hard disk was described by Thompson and Best (2000) as ‘the most important member of the storage hierarchy in modern computers’ which remains true today. The hard disk is used as internal storage in file servers, personal computers and laptops. It can be found in the form of magnetic storage and solid state, with solid state disks providing improved transfer speeds, power consumption and physical space at the expense of storage capacity. (Ekker, Coughlin and Handy 2009)

#### Future storage technologies

Current storage technologies have weaknesses when it comes to data preservation: magnetic, flash and optical media suffer from data degradation over time. While there have been comprehensive studies on the failure rate of hard disks (Schroeder and Gibson 2007; Pinheiro, Weber and Barroso 2007) showing up to a 13% annual disk replacement rate due to malfunction it is unclear how long the data on an operational modern magnetic hard disk will last before decaying but, as hard disk data density increases, the failure of part of the disk puts a lot more data at risk. More worrying are magnetic tapes, which are commonly used for backups, and optical disks are not expected to be reliable for more than five years (Simonite and Page 2010).

Data refreshing, as described in Smit, Hoeven and Giaretta (2011), protects against data degradation but requires additional effort and cost, and errors can be introduced during the

copying process. There are a number of technologies being developed which may provide a better solution: DOTS (K. Wood 2013; Group 47 2014) is a tape technology developed by Kodak and now owned by Group 47 that claims to be stable for over 100 years. Indentations are made on a non-magnetic tape and read back optically; M-DISC is an optical disc that uses materials that will allow data to last up to 1000 years (Lunt et al. 2013); The Arnano Nanoform (Rey 2012) uses engraving (lithography) to store data on a sapphire disk as a means of long term storage of several thousand years.

The memory cells that are used in flash memory are limited in the number of times they can be written before they fail (Aritome et al. 1993; Marsh, Douglass and Krishnan 1994). Magnetoresistive random-access memory (MRAM) could provide a solution to these weaknesses, providing 'relatively high read and write speeds and unlimited endurance' (Åkerman 2005) giving the potential for it to be used as computer internal memory and permanent storage.

In addition to longevity, another issue is data capacity. Hard disk drives using current perpendicular technology have a theoretical maximum capacity of one terabit of data per square inch (R. Wood 2000). This is due to the use of non-patterned media which uses grains of magnetic material deposited onto the hard disk platters. These grains are then magnetised or demagnetised to represent a bit of data. Eventually, both the size of grains and the number of grains representing single bits of data will reach a limit when the data on the disk will no longer be magnetically stable on the order of years. It is hoped that patterned media (Ross 2001), where data is stored in a magnetic array of uniform nanoscopic magnetic cells, will eventually be used to improve the data capacity of hard disks further. Self assembly fabrication methods demonstrated a potential bit size of  $50 \text{ nm}^2$  as far back as 2002 (Boardman 2005) which would give a data density of 12.9 terabits per square inch.

To better this, next-generation storage technology may use molecules (Raman et al. 2013) or DNA (Church, Gao and Kosuri 2012) to encode data, giving not only higher density but longer lifetime (Greengard 2013). Researchers have been able to encode at 5.5 petabits per cubic millimetre with DNA with the potential for data to remain readable for millennia (Church, Gao and Kosuri 2012), if not millions of years (Pääbo et al. 2004).

### **2.4.2 Local file systems**

Section 2.3.1 introduced file data as one type of computing data and discussed a number of file types. In order to store these files for later use, hardware similar to that discussed above is used, with a file system allowing the computer to manage the files and their metadata in a way that is appropriate for the device and the computer's operating system. Different file systems are more appropriate for different hardware: a USB portable flash disk will often use a different file system to a tape which will use a different file system to a hard disk. For the purposes of this section, hard disk file systems will be discussed.

The file system manages the space on the device. On a disk, when files are deleted this leaves areas previously occupied by the deleted file, so the file system allows files to be split into pieces enabling all of the free space to be used. This is achieved by breaking the disk into chunks known as blocks and keeping a record of which blocks were used by each file.

Files themselves have items of metadata that are recorded by the file system, such as its name, creation time, modification time and security information.

### **Hierarchical**

Hierarchical files systems – introduced in the Multics operating system (Daley and Neumann 1965) and discussed by the ERMA project (Barnard and Fein 1958) – allow files to be written to the storage device for later retrieval and be organised into useful groupings of directories.

### **Journaling**

Corruption can occur in a file systems if a write does not complete, such as in the case of a power loss or a crash of the computer. Journaling records the change to a journal before the write is actually performed. If there is a power cut, the computer can check the journal for changes that have not been performed correctly and replay them. This makes recovery much faster as only the journal has to be checked rather than the entire file system structure. Popular modern journaling file systems are ext3, ReiserFS, JFS and Windows' NTFS (Prabhakaran, Arpaci-Dusseau and Arpaci-Dusseau 2005).

### **Log-structured**

In a similar way to journaling file systems, log-structured file systems record changes in a log. However, in this case the log is the entire disk with file data stored as part of the log. This means that the latest changes are recorded at the tail of the log and recovery is much quicker as only the changes at the end of the log need to be checked for correctness. Checkpoints allow the last consistent state of the disk to be identified. This type of file system requires constant pruning by a segment cleaner which consolidates data split across log segments to reduce fragmentation of data and free up new segments for writing new data. (Rosenblum and Ousterhout 1992)

The advantages of log-structured file systems make them a good fit for writing to flash memory. Random access writes cause more data pages in the flash blocks to be copied and erased, reducing the lifespan of flash devices, but with log-structured file systems changes are written in a sequential manner which has the potential to extend the lifetime of the device (Min, Lee and Eom 2013).

### **Copy-on-write**

Journaling requires writing of data twice, once to the journal and then the actual change. Copy-on-write improves on this by writing new data to a new part of the disk and then updating the file system metadata to point to the new file. This makes updates faster than with journaling file systems but maintains the reliability advantages. Modern, advanced copy-on-write file systems are ZFS (Bonwick et al. 2003) and BTRFS (Rodeh, Bacik and Mason 2013).

### **2.4.3 Network file systems**

Many computers now use the internet for the exchange of data (Leiner et al. 2009) with computers connecting using wired or wireless networks. Network file systems permit the sharing of data files (stored using the technologies discussed above). Sun's Network File System (NFS) (Sandberg et al. 1985) was one early server protocol still in common use on servers; due to the popularity of Microsoft Windows personal computers, another is Microsoft Server Message Block (SMB) (Microsoft 2013b).

### **2.4.4 Distributed file systems**

Taking network file systems further, files can be distributed across multiple devices or replicated to multiple locations. xFS provided an early example of this using a decentralised architecture and a log-based file system (Anderson et al. 1996) with log segments distributed across computers and with parity to allow reconstruction in the case of the loss of a storage server. Distributed file systems have been used in data management projects: the Andrew File System (Howard et al. 1988; OpenAFS Project 2014) was used in the TR32DB data management project, and Storage Resource Broker (SRB) (Baru et al. 1998) was used for biochemistry data management. We will discuss both of these projects in Section 2.12.

### **2.4.5 Cloud technologies**

Sharing data with others is often limited by the technology available. To share an 8 GB file (e.g. a computed tomography scan in Section 2.6.5) can prove difficult: it is too big for email, personal space on institutional file servers, single-sided dual-layer DVDs and even many USB flash drives. This can require dedicated file servers to be purchased for the storage required for multiple scans which, to a small project, may not be affordable.

Cloud storage promises to remove a lot of these limitations. Examples of providers include Google, Amazon and Microsoft (Google 2010; Amazon 2010; Microsoft 2010c), although there are many more. These companies run data centres that contain tens of thousands of servers and benefit from large economies of scale (Greenberg et al. 2009), able to supply massive amounts of storage and computing power. This means that when rapidly scaling, even momentarily, cloud storage can prove to be lot cheaper (Armbrust et al. 2010); but this is an ongoing cost and also cumulative, so the costs of using these services (compared to purchasing own hardware) depends on how it is used. The purchase of a custom server requires the equipment to be specified for its maximum size and purchased outright so the initial outlay for the cloud option may be lower until storage needs grow.

Files stored in Amazon are held on multiple devices and in multiple data centres. Amazon claims to be 400 times more reliable than a typical disk drive with their Reduced Redundancy Storage service (Kumar, Lee and Singh 2012). Deelman et al. (2008) showed that cloud computing offers a cost-effective solution for data-intensive applications. Palankar et al. (2008) recommended Amazon S3 for providing high data availability and durability, noting that providing this in-house comes with high costs and engineering effort, but raised concerns

about security control for supporting scientific collaborations because Amazon S3 was lacking in terms of access control and support for auditing and delegation. It was also suggested that using local caching, such as BitTorrent (Pouwelse et al. 2005), would help improve performance and lower the costs of data transfers. BitTorrent is a peer-to-peer file-sharing system focussed on bulk data distribution (Rodrigues and Druschel 2010), making it an ideal complement to cloud storage. Due to its bartering technique, peers that have high upload rates are permitted to download at high speed, so those peers that do not contribute to the system are given lower priority.

There are many online storage providers that exist to provide cloud services (Zeng et al. 2009). Some of them use cloud providers such as Amazon and Microsoft, adding their own features with their own software, and others run their own data centres. Some of the features they provide include: huge amounts of storage space; synchronisation of files between computers; sharing of data between people; a history of files when modified and retrieval of files via web page, mobile device, software client and postage of DVD or USB hard drive in some cases.

The costs of cloud services can be difficult to estimate without previous usage data. Storage, data transfers and requests made are all charged. This is typically a few cents but the costs can quickly mount up. The transfer of large amounts of data to a cloud provider may be difficult and an institution's internal network is often a lot faster than their connection to the internet; alternatives are provided by some suppliers who will accept hard disk drives by post which is sometimes quicker than copying via the internet but there is often a charge for this (Armbrust et al. 2010).

Once data has been transferred to a cloud provider, it becomes reliant on the technology that was chosen. If the company goes out of business, valuable data could be lost, or unforeseen costs may be incurred finding a new supplier. (Armbrust et al. 2010) Backups to protect against this may involve uploading to two providers at the same time, increasing costs and complexity. Time as well as cost would be a limiting factor here.

In summary, cloud storage easily and rapidly scales: cost scales with the amount of data because you only pay for what you use and the reliability provided by some of the cloud providers defeats anything that can be achieved in-house for the cost. However, transfer of large quantities of data may be difficult, it relies on the provider chosen so if the provider goes out of business data could be lost, and backups may prove difficult (or transfer times would be increased if a secondary cloud provider is employed).

Cloud storage seems to be good in the following circumstances:

1. When a large amount of data storage is needed quickly, but for a short amount of time.
2. When the data requirements are not clearly known in advance, so it is difficult to specify exact server requirements.
3. When a suitable connection to the internet is available for the uploading and downloading of data, otherwise cloud cannot be considered.

4. For small projects who cannot afford to pay for server administrators. This is especially pertinent for short-term projects. Developers or third party solutions may still be required.
5. When reliability is a concern, but money is not available to pay for it.

This section has mainly concentrated on cloud data storage, but there is often the desire to use the data in some sort of processing. When dealing with large amounts of data, it is important to keep the data in the same place as where the processing is taking place otherwise large delays will be introduced during transfer. Many of the cloud vendors also provide processing features to operate on the data but this must be factored into the architecture and the costs before a decision can be made on where to store data.

### **2.4.6 Data transfer**

The idea of the computer reliably handling the copying of these large amounts of data is not a new one. For example, Tridgell (1999, 49) discusses the rsync algorithm which uses two signatures, one weaker but faster to compute than the other, to calculate whether files have changed and to ensure files have been copied correctly; it breaks the file into parts and uses the signatures to calculate which parts of a file have changed to avoid sending the entire file over the network. However, an ordinary user being able to share a large file easily is something that is only just beginning to be solved. Until only recently, web servers would limit files to 2 GB or 4 GB depending on the server and Internet Explorer 7 limited file download size to 4 GB (Microsoft 2009b). Other methods of transferring files such as the scp command were limited to 2.4 GB. These restrictions made it very difficult for anyone dealing with large files.

Signatures, or cryptographic hashes, such as those used by rsync are often used for verifying data integrity. There are many in common use; MD5, SHA-1 and SHA-256 (Rivest 1992; National Institute of Standards and Technology and U.S. Department of Commerce 2012) are the three most well know hashing functions. In Section 4.2.4, SHA-1 will be used to recognise whether a file has changed.

## **2.5 User requirements collection**

We now consider specific details relevant to materials engineering. For our research, collection of user requirements for the Materials Data Centre project was done in three stages. First, a questionnaire was created to assess the initial requirements and a report produced (Rimer and Reed 2009). This report was used as the foundation for future development. Second, to further understand users' data management practices, an interview was held concerning users' handling of three-dimensional synchrotron data. Finally, the initial project proposal stated that the repository would concentrate on incorporating standards-compliant formats. A series of interviews were held to look at one recognised format (MatDB) and assess its strengths and weaknesses, and establish whether mandating specific formats was the way to capture data from a materials engineering research group at a university.

### 2.5.1 User requirements report and questionnaire

A set of questions were produced to collect user requirements which are given in full in Appendix A, along with a summary of the responses. The fifty-four participants in the questionnaire were chosen to be as wide ranging as possible and included people from the University of Southampton as well as from other universities and institutions. The user requirements report listed 3 from industry, 22 academics, 9 postdoctoral researchers, 17 PhD students and 3 undergraduates. This section summarises the findings of the report and sets out the initial user requirements of the data repository.

#### Conclusions from questionnaire

The report made several conclusions which would shape the architecture of the eventual system: (Rimer and Reed 2009)

- ‘User requirements are both wide-ranging and somewhat specific to the individual research questions being pursued by the respondents, which poses challenges to the structure of the MDC.’
- ‘There is a predominantly positive response to the concept of sharing authenticated and enriched data, which can be linked to material condition (e.g. including micrographs or even fractographs).’
- ‘However shared concerns include confidentiality of data (for both sponsors and clients) and storage space requirements (particularly for images).’
- ‘There is indication that the potential value of the MDC will be linked to how user friendly it is perceived to be.’

The wide-ranging, user-specific needs, the suggestion that such a system’s value depends on its user-friendliness, and the linking of data to a material create a challenging set of requirements for a materials engineering data repository.

#### User requirements report selected user scenarios

The user requirements report selected the following user scenarios for creating a repository for materials data:

- Tensile testing (Ali, Reed and Syngellakis 2009; Sha et al. 2011)
- Long crack threshold fatigue testing (Pang and Reed 2009; Wang et al. 2012)
- Plain bend bar testing for S-N curves (Mbuya et al. 2011; Lee et al. 2009)
- Plain bend bar testing for short crack initiation and growth with replication (Mbuya et al. 2011; Lee et al. 2009)
- Fractography (Reed and Knott 1996; Aroush et al. 2006)



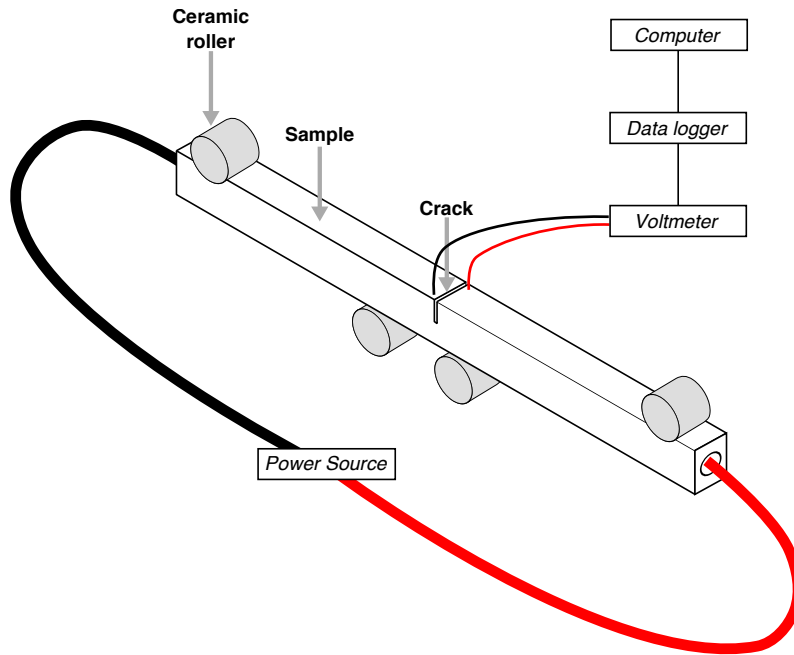


Figure 2.1: Fatigue test configuration.

The references provided are to materials engineering research produced by engineers (the prospective users of the Materials Data Centre) giving concrete examples of the data they produce and how it is used in their work.

These five user scenarios each have their own challenges. These tests will now be discussed to establish some of their associated complexities and issues.

**Tensile:** A sample of material has a uniaxial load applied to it in order to collect raw data on load versus displacement to find the stress required to cause a permanent extension to the material, which is when it stops behaving in an elastic fashion and the onset of plastic deformation begins. This is known as yield strength or 0.2% proof stress (because it is the amount of stress required to cause 0.2% extension). Other measurements such as elongation to failure and UTS (Ultimate Tensile Strength) can also be made by continuing the test to establish when the material fractures. (Callister and Rethwisch 2009, 193–95)

**Long crack growth fatigue:** Starting with a sample of material that contains a crack, the growth of the crack is monitored as loads are applied to it. An electric current is passed through the sample, as shown in Figure 2.1, and fluctuations in resistance or impedance – depending on whether direct or alternating current is being used – are detected by measuring the change in potential difference (PD) across the sample. PD increases as the resistance or impedance of the material increases, caused by the growth of the crack.

$K$  is a theoretical value representing the stress intensity of a crack based on the applied load and the crack's size and geometry. The range of  $K$  values to be applied to a sample in a fatigue test, known as  $\Delta K$ , is calculated in advance and load is adjusted to match the target  $\Delta K$  as the length of the crack changes.  $K$  values for a specimen geometry and load combination

are usually pre-existing, having previously been calculated. If they are not available, finite element analysis methods are used to calculate them.

A fatigue test begins with a pre-cracking phase or *grow-down*. Once the crack begins to grow, the load range is then reduced in order to find the  $\Delta K$  value at which the crack stops growing – the *threshold*. Often, a crack growing at less than  $1 \times 10^{-8}$  mm per cycle is considered to have reached this point (British Standards Institution 2003). In order to keep  $\Delta K$  at the required level, the load may need to be reduced as PD increases. Once the threshold has been found, the loads are increased enough to start the crack growing again, and then the rate at which the crack grows is monitored. As the crack grows, the stress intensity increases even though the loads being applied remain the same.

The raw data generated from a fatigue test consists of the PD readings in relation to time. The length of the crack ( $a$ ) is calculated from the PD voltage data and the number of loading cycles ( $N$ ) is known from the time elapsed. The rate at which the crack grows with respect to the number of loading cycles can then be calculated ( $da/dN$ ) and compared to the range of stress intensity ( $\Delta K$ ) to give a measurement of crack driving force and the material's ability to resist crack propagation at different stress intensities.

Calculating the length of the crack from a given PD value is possible because of previous calibrations with the particular material and specimen geometry. When using a material and specimen geometry where no data is available, early experiments do not necessarily yield any useful data other than data that can be used for PD calibration. It is helpful to find someone else who has already performed fatigue tests on a particular material and can give starting values for calculating crack length.

The captured data is often quite noisy so it may be necessary to smooth it during analysis. There are many methods for this, including skipping obviously erroneous sampled values or using a best-fit curve.

**Plain bend bar testing for S-N curves:** Plain bend bar testing is a different crack observation approach to long crack fatigue testing. It involves taking simple plain bend bar samples and applying stress ranges for a number of cycles to them until they fail, perhaps under special conditions such as high temperatures. This produces curve data of the stress range applied ( $S$ ) versus the number of cycles up until the sample fails ( $N$ ). When testing to a certain fraction of yield stress, tensile test data may be required.

**Plain bend bar testing for short crack initiation and growth:** From these S-N tests, surface initiating crack length ( $a$ ) and number of cycles ( $N$ ) could be calculated, and then converted to crack growth rate ( $da/dN$ ) to finally work out  $\Delta K$ . According to ASTM Standard E647 (2013), 'the growth rate of small fatigue cracks can differ noticeably from that of long cracks at given  $\Delta K$  values', which can make estimates of material lifetimes inaccurate when only considering long fatigue cracks. ASTM Standard E647 (2013) defines cracks as small when: 'their length is small compared to relevant microstructural dimension', 'their length is small compared to the scale of local plasticity' or 'they are merely physically small ( $< 1$  mm)'.

Plain bend bar testing can therefore also be used for short crack initiation and crack propagation. PD techniques used in the long crack growth fatigue test may not be possible. Instead, stopping and starting the test may be used to observe the cracks in the top surface. Micrograph data may be produced by these interruptions as optical observation techniques such as replication (applying a liquid compound that hardens to take an exact copy of the surface containing the cracks) and subsequent observation under a microscope are possible.

The crack propagation data generated in a long crack growth fatigue test can be compared to the short crack growth data.

**Fractography:** This is the study of fractures on surfaces of samples that have been tested in other ways such as a tensile test or fatigue test. A scanning electron microscope at high magnification is used to produce a set of pictures of the surface which, with detailed and skilled interpretation, can show the micro-mechanisms of failure (Parrington 2002). The conditions under which the sample was produced (and for a fatigue test, the particular  $\Delta K$  level and crack length) are required which again links back to the tensile testing data or fatigue crack propagation test. Information about the material is also required, such as composition, heat treatment information, microstructure information such as micrographs or data such as statistical distribution of grain size or precipitates in the micrographs. The images produced from these surface-observation techniques are two dimensional and further examination within the material would be destructive (Llorca et al. 1993). Methods for the examination of a failed sample in three dimensions without destroying it are discussed in Section 2.5.2.

### **2.5.2 Synchrotron data interview**

In addition to the user scenarios established by the user requirements report, some users also work with three-dimensional image data produced using microfocus computed tomography ( $\mu$ CT) or synchrotron light sources. This is used for failure analysis, like the fractography in the previous section, or to examine samples of materials or components internally and in three dimensions (Buffiere et al. 1997).

$\mu$ CT uses focussed X-rays to produce high resolution, three-dimensional images of the scanned object. Lower energy machines, up to 140 kVp, are used in soft tissue imaging applications (Yeh et al. 2009), but higher energy X-rays at high resolutions can be used in materials engineering to examine denser samples and failed specimens.

Synchrotron light sources use high intensity, almost parallel and highly coherent X-ray radiation produced by a synchrotron accelerator to examine materials to, for example, look for internal delamination splits in a sample of composite material, or find voids ahead of the crack in a fatigue test crack.

The following is information collected from interviews with Pete Wright and Anna Scott (doctoral students) about their synchrotron data and how it is managed and processed (Wright and Scott 2010).  $\mu$ CT data will be analysed in Section 2.6.5.

## About the data

To examine materials as part of their research, the interviewed materials engineers use the European Synchrotron Radiation Facility (ESRF), Swiss Light Source (SLS) and Diamond Light Source – synchrotron light sources (Bilderback, Elleaume and Weckert 2005). The data produced is three-dimensional image data. The information collected during these interviews was specifically about data from ESRF.

**Unprocessed ESRF data:** The characteristics of the data brought back by users from ESRF are listed below:

1. Average of 200 scans per collection.
2. Maximum of approximately 8 GB per scan, consisting of:
  - (a) A maximum of eight 1 GB data files containing the scan image data.
  - (b) Two information files per image (XML and a text file in a proprietary format) describing the internal layout of the image data file so the software can understand how to import the images and other metadata.
  - (c) One other small file generated by the scanner.

During the interview, it was established that this data would be copied onto a USB hard drive into a folder with a name matching the serial number of the visit (e.g. MA282), with the hard drive being stored in a filing cabinet until required. This limits the usefulness of the data to the group as the data remains offline and not protected by backups.

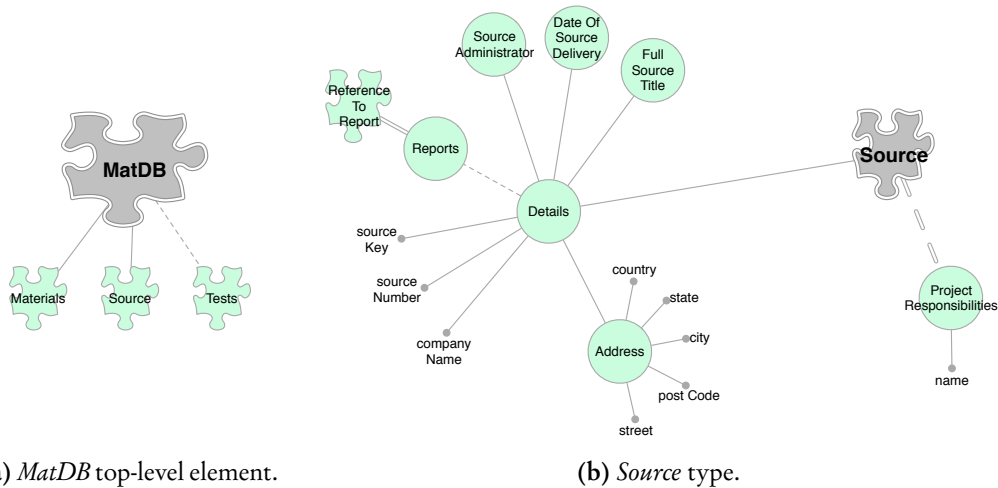
Captured X-ray projections require converting into meaningful image data using a standard back projection algorithm (a process called reconstruction), creating a three-dimensional densitometric data set (Kak and Slaney 2001, 1).

**Processed data:** The scan is loaded into software such as ImageJ or VG Studio in order to process the data. Once processed, the resulting scan data is made up of one image up to 8 GB, but on average 4 GB as not all unprocessed data will be kept because it is not of interest, contains regions which are not part of the sample or the bit-depth has been reduced to decrease the detail in order to reduce the size of the data set. The data may be processed in different ways, with the software just storing what has been done; a new copy of the data set is not required for each processing operation. The end-results may be represented as a JPEG image file for inclusion in a paper.

## Data management system feature requests

The users specifically requested the following features of a materials data management system:

- Search for data by data set creator, paper author, project name and publication title.
- The adding of data should not involve too many steps, otherwise users would not do it as they have so many data sets to upload.



**Figure 2.2:** Examples of the figures given in Appendix B, used in the MatDB XML schema interviews.

### 2.5.3 MatDB schema interview

In materials engineering, one recognised format is MatDB (Ojala and Over 2008), a standard schema able to represent data from tensile, creep and fatigue tests as XML. A set of interviews were held with Kath Soady (doctoral student) and Philippa Reed (Professor of Structural Materials) to discuss the MatDB Schema. The findings of these interviews are discussed in this section (Reed and Soady 2010).

Figures were used to help the users understand the file's structure and then discuss its strengths and weaknesses. For example, the MatDB schema consists of the following three sections as in Figure 2.2(a):

**Source section:** Details of who generated the data, as well as a link to a report.

**Materials section:** Details of the material, for example its hardness and chemical composition.

**Tests section:** The tests that were performed. Three types of tests are permitted: Uniaxial Tensile, Uniaxial Creep and Low Cycle Fatigue Strain.

A jigsaw piece was used to show where the users must refer to another diagram, for example the Source type shown in Figure 2.2(b). Using these jigsaws pieces, the user was able to build up a picture of the entire schema. The full set of figures are included in Appendix B.

#### Source section of MatDB Schema

In the Source section of the MatDB schema, the interviewees felt that Supervisor and Research Group would be useful additional data that could be recorded for use in a university. The data in the Source section contains useful data that is likely to be reused in multiple data sets but having to complete this section each time when preparing a MatDB XML file would be undesirable.

### Materials section of MatDB Schema

Both interviewees had concerns on the usefulness of the Materials section of MatDB and would not be keen on completing it. The mandatory elements of the Materials section for recording the material properties such as chemical composition and the material's name seemed sensible, but the Production element of this section contains a lot of required data which was described as frustrating by the interviewees: they felt that having something that was not so rigid would be better for their use or even free text allowing the user to record whatever they wanted. In contrast, the addition of the ability to add a microstructure image of the material being tested would have been useful.

### Tests section of MatDB Schema

In the Tests section of the MatDB schema, the interviewees found the required `TestCondition` element particularly daunting and, because there were a large amount of optional values, would want the ability to have the optional values hidden so they could be added only when required.

### Summary

Creating a MatDB-compliant data file is time-consuming for users and the interviewees felt that the MatDB data required was complex and were concerned about having to produce the XML files manually. The interviewees discussed motivations for recording such data, acknowledging that recording some of the data would be useful for future reference, would allow other people to access their data, would allow access to other people's data and would make it possible to be able to find related data.

It was felt that enforcing the MatDB standard upon users, or similar approaches such as the MatML framework, would go against the other requirement of user-friendliness even though the tests it covers were applicable to the requirements discussed in Section 2.5.1. Any MatDB files required would need generating automatically by the system to remove the responsibility from the user, but this might not always be possible if the user has not supplied the mandatory metadata which seems very likely given the comments given in these interviews. The final suggestion from the interviews would be the creation of a toolbox to allow MatDB files automatically to be created in applications used by materials engineers (Microsoft Excel, MATLAB, ImageJ) or the creation of an online web form or even a separate software application.

Based on the feedback from these interviews, support of MatDB or frameworks such as MatML is not seen as a desirable requirement for the prospective users of the MDC, but the existence of MatDB as a standard capable of supporting tensile, creep and fatigue test data is relevant, so the MDC data repository should at least support the file format for those users who do choose to use it. We will show in Chapter 3 that MatDB is supported and can be validated when publishing a MatDB data file to the data repository. When using the data repository in the manner discussed in Chapter 4 and Chapter 5, MatDB data can still be stored but it is not validated; any users wanting to use MatDB in their work can therefore do

so but mandatory data and schema compliance will not be enforced. Enhancements to enable this will be discussed in the section about plug-ins in Section 4.2.5.

## **2.6 Materials engineering data**

We have investigated a number of procedures used in the materials engineering discipline to test the properties of their materials and found a wide range of data types in use. In addition to this, we observed that there were often times in which data sets generated by one test were used in another.

The data sets generated by these tests are diverse in size and complexity. Results produced by tensile and fatigue tests can be as simple as comma separated numerical data, data sets from a synchrotron can be 8 GB in size, and computed tomography used to examine a sample produces files containing three-dimensional densitometric voxel data of up to 32 GB each – and this is likely to increase as technology improves.

This section discusses the materials engineering case studies, listing some examples of data produced and showing the relationships between different tests.

### **2.6.1 Tensile test**

#### **About the data**

Raw data is stored as CSV and is manipulated using spreadsheet software such as Microsoft Excel. The results include the yield stress, ultimate tensile strength and elongation to failure. A stress-strain plot is often drawn of the data, and geometry of the sample and images of fracture can be helpful to interpret the results.

#### **Commonly produced data files:**

1. Raw CSV data
2. Excel spreadsheet
3. Stress versus strain plot

#### **Commonly produced metadata:**

1. Specimen geometry details
2. Test date & experimentalist's name
3. Material used
4. Final values for yield stress, ultimate tensile strength and elongation to failure

#### **Dependencies identified:**

1. Material information
2. Scanning electron microscope images of fracture surface (fractography)

## 2.6.2 Long crack growth fatigue test

### About the data

Raw data is stored as CSV and its manipulation is usually done using spreadsheet software such as Microsoft Excel. Data from a tensile test is used for calculating plastic zone size, and fractography is used to analyse the cracks produced.

### Commonly produced data files:

1. Raw CSV data
2. Excel spreadsheet
3.  $da/dN$  versus  $\Delta K$  plot

### Commonly produced metadata:

1. Experimentalist's name
2. Material used

### Dependencies identified:

1. Material information
2. Tensile test results (yield stress is used to calculate plastic zone size ( $\sigma_y \rightarrow r_y$ ))
3. Scanning electron microscope images of fracture surface (fractography)

## 2.6.3 Plain bend bar testing

### About the data

Raw data is held in CSV files and manipulated in Microsoft Excel. Data from fatigue and tensile tests is useful for interpreting the results, and the tensile test results are used in calculations.

### Commonly produced data files:

1. Raw CSV data
2. Excel spreadsheet
3. Stress versus number of cycles plot
4. Micrographs produced during crack observation

### Commonly produced metadata:

1. Experimentalist's name
2. Material name

### Dependencies identified:

1. Crack propagation data from long crack fatigue test (for comparison)
2. Tensile test data
3. Material information



### 2.6.4 Fractography

#### About the data

Fractography produces images from an optical microscope or a scanning electron microscope, depending on the level of detail required.

#### Commonly produced data files:

1. Optical or scanning electron microscope images

#### Dependencies identified:

1. Test conditions
  - (a)  $\Delta K$  level (fatigue)
  - (b) Crack length (fatigue)
2. Material information:
  - (a) Composition
  - (b) Heat treatment
  - (c) Micrographs
  - (d) Statistical distribution of grain size

### 2.6.5 Microfocus X-ray computed tomography

#### About the data

$\mu$ CT produces very large data sets due to its three-dimensional nature. The data is collected from the scanner and, once converted from X-ray projections into meaningful image data (a process called reconstruction (Kak and Slaney 2001, 1)), it is processed with software such as ImageJ or VG Studio to examine the data, produce videos and select slices of interest (two-dimensional images extracted from the three-dimensional data).

#### Commonly used data files:

1. Scan image data (in the order of gigabytes to tens of gigabytes):
  - (a) Collected detector data (projections)
  - (b) Reconstructed data (raw slices)
2. Selected slices
3. Videos for illustrating raw CT image data
4. Small text files containing metadata produced by equipment

#### Commonly used metadata:

1. Scan metadata produced by equipment, e.g. exposure time, beam energy (kVp) and intensity ( $\mu$ A/mA), X, Y and Z voxel count and size

**Table 2.3:** Example metadata entries.

<b>Material Name:</b>	N18	
<b>Description:</b>	Nickel-based superalloy	
<b>Microstructure:</b>	55-57% $\gamma'$ volume fraction. $\gamma'$ solvus temperature of 1190 °C.	
<b>Heat treatment:</b>	Solution heat treatment at 1080 °C for 4 hours followed by air cooling. Ageing heat treatments at 700 °C for 24 hours and 800 °C for 4 hours, each followed by air cooling.	
<b>Composition:</b>	Element	Wt%
	Co	15.4
	Cr	11.1
	Mo	6.44
	Al	4.28
	Ti	4.28
	Hf	0.5
	C	0.022
	B	0.008
	Zr	0.019

## 2.6.6 Material information

### About the data

Table 2.3 gives an example of the metadata that can be used to describe a material. This includes information such as the material's composition and its microstructure. Images of its microstructure can also be useful. One report used at the University of Southampton is shown in Figure 2.3 which shows how this data is currently formatted by users (Thompson 2010).

### Commonly used data files:

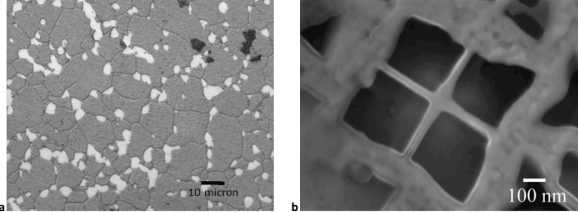
1. Micrographs

### Commonly used metadata:

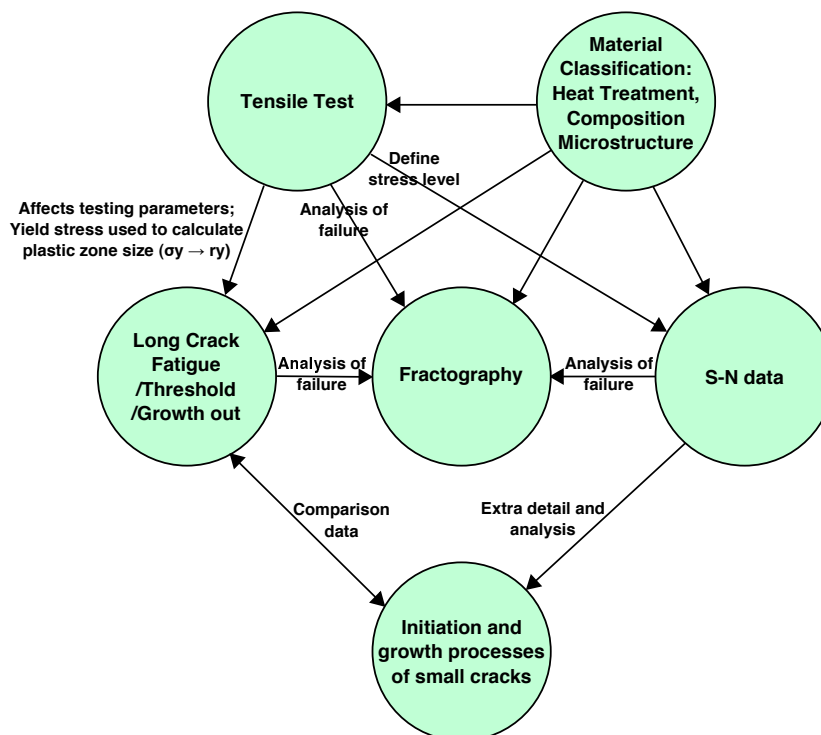
1. Composition
2. Heat treatment
3. Statistical distribution of grain size

## 2.6.7 Relationships between tests

Figure 2.4 shows how data produced in the tests described above can be used in other tests. For example, the results of a tensile test can be used to calculate how a crack will grow in a fatigue crack growth test, and a material's composition and other particulars are useful when performing or reviewing the results of most tests. The complexity of relationships between the data sets highlight that publishing a data set produced during a test is not always sufficient, and the data from other tests may also be required. Capturing these relationships can help to interpret results.

Materials Information sheet for N18									
Nickel-based superalloy									
<b>Composition:</b>									
	Co	Cr	Mo	Al	Ti	Hf	C	B	Zr
Wt %	15.4	11.1	6.44	4.28	4.28	0.50	0.022	0.008	0.019
<b>Processing route:</b>									
Powder metallurgy									
<b>Heat treatment:</b>									
Solution heat treatment at 1080°C for 4 hours followed by air cooling. Ageing heat treatments at 700°C for 24 hours and 800°C for 4 hours, each followed by air cooling.									
<b>Microstructure:</b>									
55-57% $\gamma'$ volume fraction									
$\gamma'$ solvus temperature of 1190°C									
a) Microstructure revealed by orthophosphoric etch. Grain size – 4.8-41.3 $\mu$ m.									
b) FEG-SEM image showing 400-800nm secondary $\gamma'$ and 6-30nm tertiary $\gamma'$ .									
									
<a href="http://eprints.soton.ac.uk/45815/">http://eprints.soton.ac.uk/45815/</a>									

**Figure 2.3:** Material information sheet, currently used at the University of Southampton for recording a material's composition and for describing the properties of the material. Also shows images of the material's microstructure.



**Figure 2.4:** Relationships between the different types of test data. The Materials Data Centre needs to be able to represent this.

### 2.6.8 Summary

In Section 2.5 and Section 2.6 we have discussed the user requirements report produced for the Materials Data Centre project, a data repository for hosting materials data. We have discussed the tests, and the data they produce, that will – at a minimum – need to be supported by the MDC, and highlighted the concern that the requirements are likely to be wide-ranging and user-specific, and that the user-friendliness of the repository is considered to be essential to its success. To ensure that users' requirements were continually considered, user interviews were performed to discuss use of data standards and current data management practices, and we showed that MatDB would not be a priority for the repository due to concerns of user-friendliness. Finally, we have shown that it is not just the storage of a wide-range of data that is required, but also the relationships between data sets and a large amount of metadata about each test.

In the next section, we will start to look at technologies that will help us fulfil these requirements.

## 2.7 Database technologies

### 2.7.1 Relational database systems

Relational databases are a popular approach for recording structured data. At the most basic level they can store numbers and text compartmentalised into *fields*. Related fields make up *records*, and related records are grouped into *tables*. Data in tables can then be associated with *relationships*. This model was invented by Codd (1970). Some database systems can also store more complex items such as the contents of a file.

### 2.7.2 Normalisation

Guidelines were developed to be followed when designing a database to ensure its most efficient operation. There are six sets of criteria known as *normal forms* and applying these is known as *normalisation*. Following the guidelines can help reduce repeated data and improve performance. Each normal form is cumulative, so a database has to fulfil the criteria of, for example, the first and second normal forms (*1NF* and *2NF*) in order to be considered *2NF*. (Date 2004)

Materials engineers' research data can take the form of very structured data once it is published, but other data can be very unstructured. For less structured data, there are some other approaches.

### 2.7.3 Entity-Attribute-Value

Entity-Attribute-Value (EAV) is one alternative to using an entirely normalised approach for structuring data in a relational database. EAV uses a new record for each attribute, as opposed to needing a dedicated field. It is sometimes called name-value pairs, open schema or schema-less design. (Karwin 2010)

The flexibility of this approach is that any time a new property is needed it can be added without altering the database schema. The problem with it is that EAV databases are potentially slow with large data sets and, because the database's built-in functions are no longer able to understand the data, any code or queries can be very complicated, moving the burden of work from the database engine, and administrator, to the developer.

EAV is often described as an *anti-pattern*, (e.g. Karwin (2010)) an approach that is frequently used but not considered credible due to a number of disadvantages: an inability to use mandatory values and data types; the difficulty enforcing referential integrity (because the value of an attribute is a string with no constraints like a foreign key); the difficulty ensuring attribute names are used consistently; and the complicated queries that are often needed to work with the data.

The disadvantages are generally considered to outweigh the advantages so a properly structured and normalised database with appropriate columns using the correct data types is almost always recommended. However, in certain circumstances it can be useful. Any system using EAV would have to take steps to work around the issues or ensure the disadvantages were not detrimental with the type of data being stored.

The advantages of EAV have led to some technologies arising which use key-value pairs to store data and this is discussed in Section 2.8. EAV/CR was an augmentation of EAV for use in the biomedical domain and permits the modelling of complex objects as well as managing relationships between those objects (Nadkarni et al. 1999).

### **EAV example**

To give an example of EAV, consider the design of a database for storing the material data shown in Table 2.3. There is information about the material and a list of the elements making up its composition.

**The relational approach:** Using the conventional relational approach to structuring databases, the tables 'Materials' and 'Composition' might be created, as shown in Table 2.4(a) and Table 2.4(b). The 'Composition' table would be linked to the data in the 'Materials' table by the material's key field and contain the list of elements. This is because the first normal form criteria insists that repeating data should be put into its own table.

Considering the example *Heat Treatment* data, it should be noted that there are two separate steps:

1. 'Solution heat treatment at 1080 °C for 4 hours followed by air cooling'
2. 'Ageing heat treatments at 700 °C for 24 hours and 800 °C for 4 hours, each followed by air cooling'

It is conceivable that a user may want to store these as two separate parameters which would involve making a request to the database administrator to add an extra field to the database and optionally renaming the first field. Potential field names could be 'Solution heat treatment' and 'Ageing heat treatment'. Further examination of the data reveals that the 'Ageing heat

**Table 2.4:** Relational database tables for example data in Table 2.3 under the conventional database model.

(a) 'Materials' table.

Field name	Field value
Material ID	3029
Material Name	N18
Material Description	Nickel-based superalloy
Heat treatment	Solution heat treatment at 1080 °C for 4 hours followed by air cooling. Ageing heat treatments at 700 °C for 24 hours and 800 °C for 4 hours, each followed by air cooling.
Microstructure	55-57% $\gamma'$ volume fraction. $\gamma'$ solvus temperature of 1190 °C.

(b) 'Composition' table.

Material ID	Element	Wt%
3029	Co	15.4
3029	Cr	11.1
3029	Mo	6.44
3029	Al	4.28
3029	Ti	4.28
3029	Hf	0.5
3029	C	0.022
3029	B	0.008
3029	Zr	0.019

**Table 2.5:** EAV database tables for example data in Table 2.3 under the EAV database model.

(a) 'Materials' table.

Field name	Field value
Material ID	3029

(b) 'Properties' table.

Material (Entity)	Property (Attribute)	Property Value (Value)
3029	Name	N18
3029	Description	Nickel-based superalloy
3029	Co Wt%	15.4
3029	Cr Wt%	11.1
3029	Mo Wt%	6.44
3029	Al Wt%	4.28
3029	Ti Wt%	4.28
3029	Hf Wt%	0.5
3029	C Wt%	0.022
3029	B Wt%	0.008
3029	Zr Wt%	0.019
3029	Heat treatment	Solution heat treatment at 1080 °C for 4 hours followed by air cooling. Ageing heat treatments at 700 °C for 24 hours and 800 °C for 4 hours, each followed by air cooling.
3029	Microstructure	55-57% $\gamma'$ volume fraction. $\gamma'$ solvus temperature of 1190 °C.

treatment’ data contains two stages so users may desire this field to be split, which would require another iteration of database changes. Furthermore, it can be seen that the ‘Solution heat treatment’ field involves a stage of air cooling so, in order to comply with the first normal form criteria of putting repeating fields in a separate table, at this point a new database table might be created (with the fields ‘Material ID’, ‘Treatment Step’ and ‘Treatment Description’).

Any code that has been written using the data would need modifying to follow the new structure each time it is changed and the user may need to wait for the changes to be approved through their institution’s change management processes.

**The EAV approach:** A less rigid relational model such as EAV – or even a non-relational model such as those discussed in Section 2.8 – becomes attractive in situations such as these. Under the EAV model two tables might be created with names such as ‘Materials’ and ‘Properties’, where the ‘Properties’ table has the following data:

- A link to the material in the ‘Materials’ table (the Entity)
- The property being defined (the Attribute)
- The property’s value (the Value)

Using these fields alone it is possible to record all of the required example data as separate rows in the ‘Properties’ table as shown in Table 2.5(b). Note that the ‘Materials’ table is virtually empty as everything is now in the ‘Properties’ table.

This example only covers one type of data. In a situation where there are a large number of data types, such as in the materials engineering discipline which has many types of tests – not only those described in Section 2.6 – flexibility in the database’s design becomes beneficial.

#### **2.7.4 Federated databases and GaianDB**

A federated database is a collection of independent databases that have been combined in some way to allow operations across them, perhaps by maintaining a metaschema that describes the data available for use in the federation (Heimbigner and McLeod 1985). GAIAN Database (GaianDB) (Bent et al. 2008) extends the Apache Derby relational database system (The Apache Software Foundation 2013; Zikopoulos, Baklarz and Scott 2010) to create a dynamic distributed federated database (DDFD). GaianDB installations automatically find and connect to other GaianDB nodes (vertices) on a network. SQL queries performed on one vertex are spread throughout the network and executed on all nodes that can support the query. The results are combined with the results of other nodes into a single set of results, in effect creating a single logical database. Each vertex can be configured with its own data sources, which can be Derby database tables or external sources such as other database systems, files or even internet sources (e.g. REST web service).

When a node forwards a query to its connected nodes, it is responsible for aggregating the results into a single result set before passing the results back to the node that sent it the query.

**Table 2.6:** Some of the structured storage systems.

System	Description
CouchDB	Document database (JSON). JSON permits strings, numbers and lists.
MongoDB	Document database (BSON). BSON adds to JSON by also permitting dates and binary data. The Drupal content repository can use MongoDB for the storage of field data.
Neo4J	Graph database. Can store nodes and relationships between nodes. Properties of the nodes and relationships are stored as key-value pairs.
Midgard	Content management framework. Parameters can be entered as a triplet of strings: parameter domain, parameter name and parameter value.
Jackrabbit	Content repository. This is an implementation of The Content Repository for Java Technology (JCR) API Specification which is an abstract model for accessing a content repository. Uses a hierarchy of nodes and properties. Nodes can have types and these types specify the properties that are available for each node and any child nodes.

Eventually, the originating node will receive and aggregate the results from its connections, giving a set of data that represents all nodes on the network that could answer the query.

The GaianDB project has developed algorithms for efficiently adding nodes to a network and is investigating various approaches for modelling and maintaining the network graph.

## 2.8 Structured storage

A number of technologies exist that permit storage of documents and tagging of metadata. Enterprise document management systems (that permit storage and tracking of documents) and web content management systems (that permit users to provide web page content and have the system automatically generate the web page) are examples. Both of these types are driven by the content, where the files or the web site are the basis of the system. The term ‘content repositories’ is used to describe this approach (Hubmann-Haidvogel, Scharl and Weichselbraun 2009).

These systems take a variety of approaches to storing the data to provide flexibility to their users. Many of them try to move away from using a standard relational database which is no longer a good fit – this is called structured storage or sometimes NoSQL (Rys 2011; Meijer and Bierman 2011).

Some of the underlying data storage mechanisms include: *document databases*, where a document is a collection of fields with associated values and each document’s set of fields is permitted to differ; *graph databases*, which can represent relationships between entities with the ability to record data as key-value pairs on the entities and the relationships; *key-value stores* which can look up the value of something based on an identifier; and *object databases* which are commonly used with object oriented languages and permit a more natural mapping of data objects held in memory to permanent storage.

Examples are shown in Table 2.6. CouchDB (Anderson, Lehnardt and Slater 2010) is an example of a document database. Values of fields in each document are permitted to be strings, numbers, dates and lists of values. MongoDB (Chodorow and Dirolf 2010) is another example and supports a lot more data types. It uses BSON (Chodorow and Dirolf 2010; *BSON specification* 2011) as its data storage format, a binary format based on the JSON (Crockford



2006) text-based format (used to represent data structures in JavaScript), and supports all of the JSON types and introduces additional ones such as dates and binary data. The Drupal content repository (Byron et al. 2008) can use MongoDB for the storage of field data. By default, the flexibility of allowing custom fields to be created is accomplished by creating a new table in an underlying relational database for each new field required. When using MongoDB for storage of field data, Drupal uses one document containing all of the values of the fields, which can improve performance for massive data sets as it avoids the joining of data from many tables.

Neo4j (Dominguez-Sal et al. 2010; The Neo4j Team 2011) is an example of a graph database which can store nodes and relationships between nodes. Properties of the nodes and relationships are stored as key-value pairs. Midgard (Midgard Project 2011) is another example of data storage that does not impose a schema; parameters can be entered as a triplet of strings: parameter domain, parameter name and parameter value. This is similar to the entity, attribute, value approach of EAV.

Most of these categories tend to use some form of key-value pair arrangement. Documents held in a document database are made up of key-value data. The properties stored in graph databases take the form of keys and values and key-value stores obviously use this model. EAV data has similarities to key-value data, as the attribute name could be used as a key to the finding the value.

### **2.8.1 The Content Repository for Java Technology API specification**

The Content Repository for Java Technology (JCR) API specification (Nuescheler 2005, 2009) is an abstract model for accessing a content repository. Jackrabbit (The Apache Software Foundation 2011) is one implementation of this type of repository but there are others. The architecture uses a hierarchy of nodes and properties. Nodes can have types and these types specify the properties that are available for each node and any child nodes. For example, the `nt:file` node-type specifies a sub-node called `jcr:content` must exist which can be used to store the contents of a file as a sub-property.

The Jackrabbit implementation permits files to be stored in a file system or database. When using a file-based store, the limitations of the database are removed, but files are given a name that is generated by a hash of the content. This has the advantage of removing the possibility of duplicated files, but it also means that users must manage their files and metadata through the repository's interface. This is because the names of the files are meaningless when viewed directly in the file system. For the Materials Data Centre, we wanted users to be able to save and retrieve files without needing any special software or training, so this was a disadvantage for us.

## **2.9 Document repositories**

### **2.9.1 Institutional document repositories**

EPrints (Gutteridge 2002), DSpace (Bass and Branschofsky 2001) and Fedora (Payette and Lagoze 1998) permit storage of publications and can be adapted for data as shown by the

eBank project which established the eCrystals repository (Coles et al. 2006) that managed and disseminated metadata relating to crystal structures and investigated linking data sets from Grid/Cloud-enabled experiments to open data archives and through to peer-reviewed articles using aggregator services. This allowed crystal structure data to be provided with a paper for a reader to check validity.

Data Dryad (Dryad 2014) is a good example of a data archival repository that is built using DSpace. DSpace stores files on the file system or can integrate with Storage Resource Broker (SRB), a distributed file system. The API or the interface must be used to upload files into the file store as DSpace gives each file a unique name that it generates itself. Files cannot be loaded directly into the file store as the system will not recognise them. This is a similar approach to Jackrabbit (The Apache Software Foundation 2011) where names of files are generated using a hash of the content removing the possibility of duplicated files. This means that users must manage their files and metadata through the repository's interface.

DataStaR (Dietrich 2010) uses Fedora for data set storage, with their metadata being managed by Vitro which specialises in semantic metadata management. Gutteridge (2010) also investigated data set storage within a document repository. Treloar, Groenewegen and Harboe-Ree (2007) discusses migration of data from data collaboration repositories into institutional repositories across a notional 'Curation Boundary'. Repositories before this boundary are often referred to as a local repository, collaboration repository or data staging repository.

Microsoft Zentity (Carr et al. 2009; Singh, Witt and Salo 2010) permits the storing of authors, papers, data, videos, code, lectures, books and other entities in a database and then provides the ability to make relationships between them. In many ways, it is similar to a document repository by permitting the storing of papers and data about authors, but the way the data is handled is different as an author must be configured in the system and a paper linked to it in a manner similar to a relational database.

We will show in Chapter 3 how the EPrints document repository system can be extended to link to an external data repository.

### 2.9.2 Microsoft SharePoint

Microsoft SharePoint has some document management features and also has the advantage that it allows metadata to be stored, and provides social networking features which might make the MDC more useful to its users. SharePoint provides a framework upon which to build an application. Out of the box it already has powerful search capabilities, many of the desired social networking features such as blogs and wikis, user friendly web page creation as well as basic document management and workflow, and can be extended with plug-ins known as web parts.

Chapter 3 will show Microsoft SharePoint used for data storage. There are other products with similar functionality available but Microsoft SharePoint was most supported at our institution. The size of file that can currently be stored by SharePoint is limited to 2 GB (Microsoft 2013c, 2014e) – a remnant of its use of Microsoft SQL Server for its data – which will lead

to the Materials Data Centre in Chapter 4 using a different approach for file storage with interface components still within SharePoint.

## **2.10 Protocols and frameworks**

To support interaction with the document repositories discussed in Section 2.9.1, the following protocols and frameworks are used: SWORD; OAI-PMH, OAI-ORE and RDF. This section provides a description of these.

### **2.10.1 RDF (Resource Description Framework)**

RDF (Lassila and Swick 1999) is a family of specifications designed to allow the relationships between objects to be defined. It consists of a set of statements, known as a triple, with three parts: the subject, the object and a predicate. The predicate describes the relationship between the subject and the object.

RDF is an abstract model: RDF relationships can be stored in a database known as a triplestore or it can be described using XML with RDF/XML (Gandon and Schreiber 2014). Additional vocabulary can be defined with the RDF Schema language (RDFS) (Guha and Brickley 2004) or The Web Ontology Language (OWL) (Schneider 2009). OAI-ORE, described below, uses RDF/XML to describe the resource map of objects in a repository.

The advantage of RDF is its flexibility. Given any number of objects, the relationships between them can be easily described without the database needing to be redesigned in contrast to the approach a relational database takes where the structure is very rigid.

### **SPARQL**

Tools that can understand the RDF format include SPARQL (Prud'hommeaux and Seaborne 2008) which is a language for querying RDF graphs and can be used to perform advanced searches with the data.

### **2.10.2 Atom Publishing Protocol and SWORD**

The Atom Publishing Protocol (Gregorio and hOra 2007), better known as AtomPub or APP, is a protocol that uses HTTP (Fielding et al. 1997) for creating or updating web resources. SWORD (Allinson, François and Lewis 2008; SWORD Project 2009, 2013) builds upon this standard to provide a method of depositing into a data repository. Rather than depositing an Atom document as with AtomPub, SWORD sends files. SWORD version 2 added updating and deleting of resources – functionality which was not included in earlier versions of the protocol (Allinson et al. 2008; Jones et al. 2012).

### **2.10.3 OAI-ORE (Open Archives Initiative Object Reuse and Exchange)**

The OAI-ORE specification (Lagoze et al. 2008) makes it possible to represent a collection of resources. OAI-ORE can represent a set of pages that go together, which helps search engines

identify groups of web pages rather than requiring clever heuristics. A resource map describes each resource in this aggregation, expressing the relationships and properties. It can be given in different formats such as RDF/XML and Atom XML.

#### 2.10.4 OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting)

The OAI-PMH framework (Lagoze et al. 2002; Sompel et al. 2004) provides a method of contacting a repository and querying it to find out about its records. It supports the instructions: `GetRecord`, `Identify`, `ListIdentifiers`, `ListMetadataFormats`, `ListRecords` and `ListSets`. The `metadataPrefix` argument is used to specify what format the results should be returned in (such as Dublin Core, a standard way of structuring metadata).

## 2.11 Accurate data capture

Metadata is used for describing, categorising and linking data, and complete and accurate metadata can be useful for data discovery. Appropriate metadata will also assist with data preservation concerns introduced in Section 2.3.1, particularly locating and understanding the data. Some of the concepts and technologies related to this will now be discussed, followed by an introduction to the Semantic Web which also relies on accurate data and metadata collection.

A *controlled vocabulary* can ensure accurate metadata by defining the preferred terms to be used within a system. If all metadata uses an agreed set of terms, data set discovery and data sharing can be improved if all users know and use the terminology. However, fixed vocabularies can cause issues for some. For example, Novotny (2004) showed that people demonstrate an inability to use synonyms in order to ‘repair’ their searches in order to match their vocabulary against the terms used by a system. Antelman, Lynema and Pace (2006) highlighted that people have large but inflexible query vocabularies and discusses the use of *entry vocabularies* to match a user’s vocabulary against the vocabulary in the repository.

Due to the complexity of the vocabularies of some disciplines, it is not just a case of listing a set of terms that can be used within a system as these terms often have a complex relationship with one another. A *taxonomy* organises terms into a hierarchy, and *thesauri*, like taxonomies, allow terms to be related hierarchically but also can define similar and associated terms (Daconta, Obrst and Smith 2003, 159). A thesaurus can be useful to the user who might not know the terminology in the controlled vocabulary and it can also help the system to relate terms used to those held in the system.

An *ontology* is a formal way of describing knowledge, such as controlled vocabularies, thesauri and taxonomies, in a hierarchical way. There are a large number of languages for constructing ontologies, the two most common two being OWL and RDF Schema (Cardoso 2007) both of which can be represented using RDF/XML introduced in Section 2.10.1.

### **2.11.1 Semantic web**

These techniques and innovations are among those that underpin the *Semantic Web*, a term employed by the World Wide Web Consortium (W3C) to describe its vision of using the internet to share structured data via standard formats and protocols. Thorough use of controlled vocabularies, accurate recording of relationships between data sets – and the meaning of those relationships – and a common framework that can be understood by computers has the potential to allow data exchange across disciplines and institutions.

Data published on the internet using accepted standards is referred to as *Linked Data* by the W3C. A five star ratings scale suggested by Berners-Lee (2006) and adopted by the United Kingdom Government to measure the usability of its Open Data initiative (Cabernet Office 2012) describes the openness of data in a system. Publishing any data on a web page on the internet with an open licence equates to the first star on this scale. If this data is in a machine-readable, structured format (e.g. Microsoft Excel file containing data as opposed to tables or graphs stored as images) then a second star is awarded. Three star data means that the data is stored in an open, standardised, non-proprietary format such as CSV or XML. Four stars are achieved if W3C approved open standards such as RDF are used by the archive so that others can link to the data and the fifth star is when there are contextual links to other people's data.

### **2.11.2 Discussion**

This section has introduced controlled vocabularies, taxonomies, thesauri, ontologies, and the Semantic Web and its use of these concepts and other technologies. Taxonomies allow a formal definition of metadata terms and their organisation into a hierarchical structure. As shown in Section 2.5, users require flexibility from the system and are less concerned with formal standards, deterred by the rigidity of some. In addition, ontologies for many of the materials tests have not yet been established. To give meaning to materials data, there is a lot of metadata that must be provided and the goal of the system presented is to handle a wide range of data and metadata, as well as to be amenable to unknown data and metadata types. While controlled vocabularies hold many attractions in the world of published data, it is less appropriate to a user-driven system with an unclear set of data that is yet to be published. For the same reason as choosing EAV over a relational database, flexibility without the involvement of an administrator is the overriding priority here in order to capture data and metadata that might otherwise be lost.

A miscellaneous, free-text field might solve the problem if a user wants to deviate from required metadata defined in a taxonomy but name-value metadata pairs will prove more useful even if the terms used do not match any formal standard. There is also the question of what a user must do when mandated metadata is not yet available – to the user, this is a perfectly valid scenario that would not fit with formal standards used in the world of published data. A less formal, unstructured approach, where the user is control of their metadata terms is often called a *folksonomy*. This approach fits the system's requirements better. Therefore, the system presented allows users to control their own metadata but provides features that help to improve metadata quality which will be discussed in Section 4.7.2.

There are many useful technologies that we have already discussed in this chapter that support the Semantic Web: Unicode for text encoding, XML for the syntax to represent a document's structure, RDF for data interchange, and SPARQL for querying. Support and validation of MatDB XML data is shown in Section 3.2.2, and RDF and SPARL will be explored in Section 5.2.8. The use of these technologies show the possibility of the system eventually participating in the Linked Data model, although its unquestioning acceptance of any data would earn it only one star on the scale introduced in Section 2.11.1.

## 2.12 Data management projects

We now briefly survey a few data management projects from different domains. Systems with such wide-ranging sets of data have been attempted in the biomedical domain. EAV was used in the early 1980s for medical record management where doctors could define their own data dictionaries to track diagnoses and treatments of patients (Stead, Hammond and Straube 1983). The history of EAV is comprehensively described in Dinu and Nadkarni (2007). EAV was then augmented with the *EAV with classes and relationships* (EAV/CR) approach which permits the modelling of complex objects as well as management of relationships between those objects (Nadkarni et al. 1999; Marengo et al. 2003; Jäger et al. 2009).

Work in other research domains such as chemistry with eCrystals (Coles et al. 2006) and biology with OME (Linkert et al. 2010) indicate that a Materials Data Centre would be valuable, but the breadth of data types make its architecture a challenge.

The LabTrove electronic notebook system (Frey 2008) allows users to connect sources of data to a blog. Automatic publishing of data from a testing machine might also be possible as the project has begun looking at monitoring a file share and automatically creating blog entries. The system is mature and could be used for encouraging the sharing of data within an institution.

The approach taken by the BioSimGrid project (Ng et al. 2006) to promote data sharing between biochemists used a distributed file store – specifically SRB – combined with a relational database. Data and files were replicated across participating sites in Europe and the United States. The structure of the BioSimGrid database was solely for biomolecular simulation metadata; MDC, in contrast, will be used to store very general data.

The LOFAR project (Begeman et al. 2011), adapting the Astro-WISE system (Mwebaze, Boxhoorn and Valentijn 2009), permits users to store large files in a file system and then access them via the HTTP protocol and permits metadata, including relationships between objects, to be recorded in a separate database. The system uses Grid technologies for processing the data and so it can scale across multiple sites. Many operations, such as data upload, are performed with supplied tools in the Python programming language.

Similar architectures have been created in the Earth Science and Environment domains (Fiore, Negro and Aloisio 2011) built upon distributed collaborative technologies (Foster et al. 2003). Ensuring that the MDC could later be adapted to use a similar approach seems necessary to allow it to expand to multiple sites.

The convenience of using a portal to browse stored data before downloading it, saving on time and bandwidth, was demonstrated by Pallickara, Pallickara and Zupanski (2012) with the Atmospheric Data Discovery System. The system, which processes and indexes meteorological data in BUFR format from the atmospheric science domain, permitted users to publish the BUFR metadata and query and search over the data sets and browse the metadata through a web browser before downloading.

TR32DB is a data management system created for the Transregional Collaborative Research Centre 32 to manage data created by the project along with accurate metadata (Curdt et al. 2012b). The project itself is multi-disciplinary, including data from soil sciences, meteorology, geo-informatics and mathematics, with many data formats from a few kilobytes to several gigabytes. Due to the wide-ranging data types, a file system (the distributed file system, the Andrew File System) was chosen for storing data, and a metadata database and web interface are used for the addition of metadata relating to any uploaded files. Metadata is strictly controlled in order to meet metadata standard requirements (Curdt et al. 2012a); it can be added one day after files have been uploaded and any additional metadata keywords not defined are requested through the system administrator (Transregional Collaborative Research Centre 32 2014). Files are moved out of staging folders into administrator-controlled folders in the data management system by an overnight script, which prevents data from being modified once it has been uploaded (Curdt et al. 2008).

The Open Microscopy Environment (OME) project in biology (Linkert et al. 2010) identified the need to share multidimensional biological microscopy image data (Linkert et al. 2010) and developed software and protocols allowing image data from any microscope to be stored, shared and transformed without loss of the image data or information about the experimental setting, the imaging system or the processing software (Goldberg et al. 2005). It provides a data model implemented using a PostgreSQL relational database (as detailed in the documentation (University of Dundee & Open Microscopy Environment 2014)) where metadata is stored about files in a file system, and defines an OME XML file format to permit data to be exchanged with other OME databases. The OME XML schema supports the storing of the image data, experiment metadata and results. The database stores the metadata about the image acquisition and any processing and analysis done. The subsequent OMERO project introduced a file system monitoring tool to recognise new files in a specific folder and automatically import them into the OMERO file system used for storing binary files in OMERO (Allan et al. 2012).

### **2.12.1 The Materials Atlas (materials data repository)**

An existing materials repository, the Materials Atlas (The Minerals, Metals & Materials Society (TMS) 2013) project, sponsored by the Office of Naval Research (ONR) and the Defense Advanced Research Projects Agency (DARPA), uses collaborative software to collect materials data, experiments and simulation data sets, along with information on the software tools that users of the site may find relevant. The site is powered by Atlassian Confluence – commercial software that provides wiki-like pages, file sharing and other collaborative features. Atlassian

Confluence currently supports files up to 2 GB (Atlassian 2013). Users log in and create pages from a template for the type of data they want to store, uploading and attaching data files to the web page and providing metadata by completing the template.

### 2.12.2 Human Genome Project

The Human Genome Project was a 15-year international project to identify all nucleotides in human chromosomes, with up to 100 000 genes each having up to 1 million nucleotides (Sargent et al. 1996). This is a lot of highly complex data. For example, GenBank (Benson et al. 2010), the European Molecular Biology Laboratory (EMBL) Nucleotide Sequence Database (Kulikova et al. 2004) and the DNA Databank of Japan (Kaminuma et al. 2011) are public databases containing nucleotide sequences and bibliographic metadata of more than 300 000 organisms. This includes 255 billion nucleotide bases (Benson et al. 2010).

Molecular biology data banks, as with the MDC, contain diverse data and a common approach in this field is to store it in a flat text file. This has the advantage that it can easily be read by humans or parsed by software tools such as SRS (Sequence Retrieval System) for indexing and querying (Etzold, Ulyanov and Argos 1996). Each unit of data, such as the description of a disease or the coordinates of a protein atom, on its own is small, but collectively represents a massive amount of data; the EMBL database at present is 2.1 TB (EMBL Outstation – The European Bioinformatics Institute 2014).

AceDB (Stein and Thierry-Mieg 1999) provides an object-oriented database management system and a set of components that provide a framework for querying data and rapidly building applications that use the data.

Newer projects are moving to relational databases to improve performance when performing their analysis such as the Ensembl system (Potter et al. 2004).

In order to record the many different data types and their relationships, the Utah Center for Human Genome Research developed a database which would allow relationships between objects to be stored as data in a database in contrast to modelling these relationships in the database schema. This had the advantage of reducing the number of schema changes to the database as the relationships evolved (Sargent et al. 1996).

### 2.12.3 Large Hadron Collider

The Large Hadron Collider (LHC) at CERN is a two-ring superconducting hadron particle accelerator and collider in an almost circular 26.7 km tunnel made up of eight straight sections and eight arcs (Evans and Bryant 2008). It has a number of sensors each looking for different events: there are four large detectors for the experiments known as ATLAS, ALICE, CMS and LHCb and two smaller ones for TOTEM, with detectors either side of CMS, and LHCf, with detectors either side of ATLAS. ALICE (A Large Ion Collider Experiment), for example, is watching for large ion collisions (The ALICE Collaboration et al. 2008), while LHCf is researching the secondary particles produced by collisions in the larger ATLAS experiment (The LHCf Collaboration et al. 2008).



The LHC produces massive amounts of data: the four main detectors produced 13 petabytes of data in 2010 (Brumfiel 2011) and the different experiments of the LHC have to cope with gigabytes per second (Toledo, Mora and Müller 2003). The ALICE experiment is required to process data at 1.25 GB/s transforming the data into files and storing it in Permanent Data Storage (PDS) ready for processing (Divià et al. 2010). The *ALICE Environment for the Grid* (AliEn) is a framework that is used by ALICE scientists to process data on Grid computers. The AliEnFS feature (Peters, Saiz and Buncic 2003) of the AliEn Grid service provides the ability for tagging files with metadata. The user creates text files specifying the details of the metadata to be recorded and associates it with one or more folders. The system then automatically creates a table in a relational database with the columns specified and then the user is able to specify metadata values for any file under that folder (The ALICE Collaboration 2011). This provides tremendous flexibility: folders can be associated with multiple metadata tables and metadata tables can be associated with multiple folders.

Oracle relational databases are also used to support the LHC for recording data such as detector conditions, calibration and geometry, and are fundamental to supporting the separate physics data collection processes; even these have 300 TB of disk storage (Girone 2008). A support system, the ALICE Electronic Logbook (Altini et al. 2010), permits the ALICE project's users to record metadata about their activities. The user can choose from predefined fields and can generate reports with the ability to attach any type of file. The system uses a MySQL relational database and contains nearly 4 GB of metadata.

## 2.13 Summary

In this chapter, we have examined the various tests that a materials engineering data repository would, at a minimum, need to cope with. We have also discussed types of computing data and technologies used for its storage, and discussed some projects that have had to cope with the management of data. Some of these approaches and technologies have provided inspiration for the final system discussed in this thesis: eCrystals produces a single summary page for each sample that has been entered; the data blogging concept used with LabTrove is useful and one of the reasons Microsoft SharePoint was considered because it provides blogs, wikis and message boards useful for groups of users to coordinate; the extra layer above EAV that EAV/CR introduces provides a structure that is not present with plain EAV and this helps to avoid some of the disadvantages of EAV – a similar approach has been taken with the database discussed in Chapter 4 in order to have flexibility but also some structure.

In the materials tests we have considered, there is a variety of data files, including CSV encoded raw data, Microsoft Excel spreadsheets, two-dimensional and three-dimensional images, and videos. The metadata produced includes data about the test performed, key results of the test and information about the material upon which the test was performed. The variety of data set types, sizes, metadata and relationships between data sets create a complex problem in terms of storing, managing and retrieving. For systems where the data is clearly defined, the standard approach to designing a relational database system makes sense. For example, the results from a standardised tensile test and some information about the material tested would

be a suitable candidate for a relational database. However, the MDC is required to support a wide range of data types across the materials discipline, including those use cases discussed in Section 2.5 and Section 2.6. Designing a rigid structure to cover all types of metadata where it is not always clear how the user will organise their research data or even what form it will take is an unbounded task, and this becomes even more difficult in the event of a cross-discipline project.

Many databases impose size limits on the files they store, such as the 2 GB maximum file size imposed by Microsoft SQL Server 2005 and earlier, and the 4 GB boundary in MySQL (MySQL Documentation Team 2011, 798). We saw this in the Materials Atlas in Section 2.12.1 and Microsoft SharePoint in Section 2.9.2. We will discover that this proves problematic for larger data files, such as 3D voxel densitometric data sets, in Chapter 3. Even for databases without this limitation, file storage in a database introduces a level of complexity which could make recovery in the case of a disaster more difficult.

Many of the solutions provided by structured storage differ from the standard relational database in that they do not impose the use of a rigid schema. They tend to use some form of key-value pair arrangement: documents held in a document database are made up of key-value data, and the properties stored in graph databases take the form of keys and values. EAV can easily be converted to a key-value approach as the attribute name can be used as the key to the finding the value which maps well onto key-value technologies such as Amazon's SimpleDB and DynamoDB (Amazon 2013; DeCandia et al. 2007), Microsoft's Azure Tables (Mizonov and Manheim 2013) and Google's AppEngine Datastore (Chang et al. 2008), which are Cloud database services and permit scaling to massive levels.

Much of the metadata for this project can be stored with attribute and value fields, but the nesting of metadata in our materials information example given in Table 2.3 suggests that attribute and value fields may be insufficient to support all metadata. We will show in Chapter 4 the use of EAV for metadata storage, but to fully support our requirements we enhance it by adding extra fields and tables, making our final solution a relational database approach with an extended EAV table for storing the metadata.

The next chapter will look at how data can be published with the EP2DC plug-in and will show an early prototype of the materials data repository.



## Chapter 3

# Data publishing with EPrints


---

The work described in this chapter has been presented as:

Tim Austin, Mark Scott, Steven Johnston, Philippa Reed and Kenji Takeda. 2011. 'EP2DC – an EPrints module for linking publications and data'. In *Proceedings of PV 2011: Ensuring Long-Term Preservation and Adding Value to Scientific and Technical Data*. Toulouse, France, 15th November

---

### 3.1 Introduction

 HE MATERIALS DATA CENTRE is concerned with the capture of data whilst it is still in use, but at the end of its life cycle the data may be used to create a publication so, to support our objective of integrating the system with established document publishing technologies, we must first consider the publication of data. In this chapter, we discuss EP2DC (Reed et al. 2009; Austin et al. 2011), a JISC funded project to produce a plugin for EPrints to permit data to be transmitted to a data centre at the same time as a paper is deposited. The project proposal specified that:

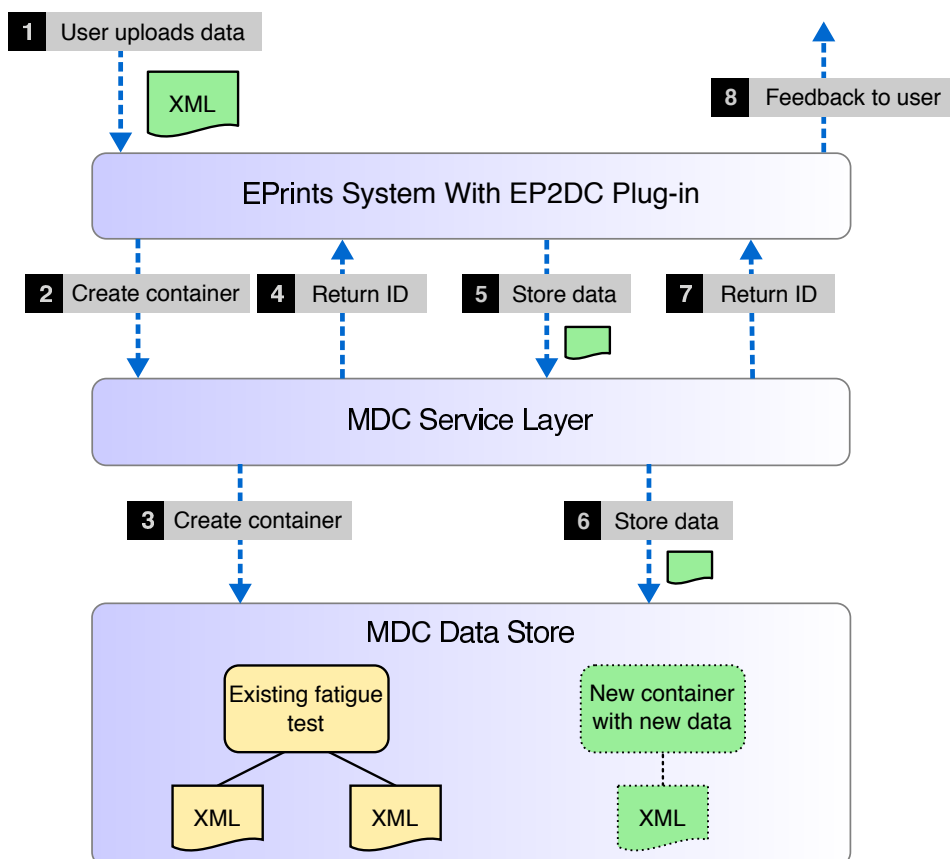
The objective is to develop a prototype module to enable the EPrints repository to support the submission of XML-formatted experimental data together with the manuscript to which they correspond. The basic motivations for the work are to promote the conservation of experimental data and to link data to publications. (Reed et al. 2009)

The project involved setting up a prototype Materials Data Centre to host the data, writing a service layer with which EPrints could communicate and customising EPrints so that data in the data centre could be accessed. Due to the nature of the data that the data centre must handle (published data), the end product was designed to accept XML data files which would only be allowed if they validated against their schemas.

Weaknesses found with this MDC prototype would eventually lead to the model we present in Chapter 4, but the EP2DC project lays foundations for the future MDC to support data at the end of its life cycle and shows how publications can be linked to data.



**Figure 3.1:** EP2DC Deposit Workflow. The ‘EP2DC Data’ step is handled by the EP2DC EPrints plug-in.



**Figure 3.2:** Components in the EP2DC architecture and an illustration of the conversation between EPrints and the remote data centre when a data file is deposited.

## 3.2 Architecture

The EP2DC project customised an EPrints system to enable it upload data to a data centre, inserting an extra step into the standard EPrints workflow to upload data at the same time as a publication. As can be seen in Figure 3.1, this is after the upload of the publication which occurs in the ‘Upload’ step. Other steps remain unaltered, so mandated publication metadata that is usually collected in the other steps still occurs.

Figure 3.2 shows the components in the EP2DC architecture and illustrates the conversation between EPrints and the remote data centre when a data file is deposited. The data centre’s data store will be described in Section 3.2.1 and the service layer with which EP2DC communicates is discussed in Section 3.2.2. The enhancements made to EPrints are detailed in Section 3.2.3.

### 3.2.1 Early MDC prototype

The first component we discuss is the remote data centre itself, whose only requirement for this project was to store published materials engineering test data; the chosen format for this was MatDB which is the XML file format discussed in Section 2.5. Although we have shown in Section 2.5.3 that MatDB is not a priority for the Materials Data Centre, the use of MatDB for testing EP2DC ensures its support and will allow us to demonstrate validation of data files.

Windows SharePoint Services 3.0 was chosen for the data storage component as it provided the ability to store and share files easily and adds other features discussed in Section 2.9.2 which may prove beneficial when developing the interface features of the data centre. For this project, we will only store MatDB test data, but SharePoint will support other file types which provides a degree of future proofing. The University of Southampton, as well as many other institutions, have central IT infrastructure using this product so it was a good choice for us.

SharePoint web sites are stored in *site collections* which are created using a template. For this project, we used the ‘Blank site’ template. Other templates may be used to gain additional SharePoint features such as announcement lists, calendars and contact lists but these were not desired for this version of the MDC.

SharePoint stores documents in *document libraries* and distinguishes between types of file with *content types*. Content types also allow SharePoint to collect metadata specific to a file type which promised the ability for collecting extra metadata and building additional features such as identifying related files. Two document libraries, given the names *mdcdata* and *mdcschemas*, were created for storing the files passed by EPrints and for holding XML schemas that were not already on a publicly accessible site. A custom content type, called *MatDBTest*, was required to distinguish between ordinary files and materials tests, and was based upon the built-in Document content type with all the same metadata.

### 3.2.2 Service layer

The data centre’s web service, which is used by EPrints for data set upload, was written in the C# programming language and utilises the Windows Communication Framework in Microsoft .NET Framework 3.5. It is a RESTful service and therefore follows the constraints set out in Fielding (2000).

Using the service, the EP2DC plug-in can upload files, request information about a file such as its download URI and list files that are in a data set. The web service can also be queried for other data items that might be related, which is returned as HTML for embedding in the EPrints web page; we return the filenames of the items and their download location, but this could be any information available such as embedded images or any useful feedback that could be generated by the data centre based on the file that was uploaded. We will show this in use in Section 3.3.

Table 3.1 details the operations that are possible via the EP2DC web service layer.

**Table 3.1:** List of data centre operations made available via the REST service.**CreateContainer** `http://mdc/containers/`

This operation creates a container inside the data centre for file upload. When viewed in the data centre, this is shown as a folder. It returns the identifier of the created container and is activated via HTTP POST.

**UploadFile** `http://mdc/containers/{MDCcontainerID}/objects?fn={fileName}&length={`  
`↪ length}&validateXML={validateXML}`

Once a container has been created, a file can be uploaded into it with this operation. A flag is set if the file is an XML file and validation is required. If the validation is successful, the identifier of the uploaded file is returned, otherwise the file is rejected and an error message is embedded in the response to EPrints so the user can be notified. It is accessed via an HTTP POST request.

**AddMDCIDToContainer** `http://mdc/containers/{MDCcontainerID}/linkedobjects?mdcid={`  
`↪ mdcid}`

This operations links a file that is already in the data centre to a container. It returns an identifier and is accessed via an HTTP POST request.

**AddURIToContainer** `http://mdc/containers/{MDCcontainerID}/links?uri={uri}`

This links to a file that is outside of the data centre to an existing container. It returns an identifier of the object that was created and is accessed via an HTTP POST request.

**GetHTMLForMDCID** `http://mdc/allitems/{MDCID}/info/html`

Given the ID of a file, this operation generates some HTML which EPrints embeds in its web page. The prototype lists files of the same type. It is accessed via an HTTP GET request.

**GetMDCIDForObjectPath** `http://mdc/getmdcid/?object={objectPath}`

The prototype data centre tracks objects using MDC IDs which are GUIDs. If the MDC ID is not known for a particular file, it can be retrieved using this operation by passing the object's path. It returns an identifier and is accessed via an HTTP GET request.

**GetMDCIDsInContainer** `http://mdc/containers/{MDCcontainerID}/`

This lists all of the identifiers of files within a container. It is accessed via an HTTP GET request.

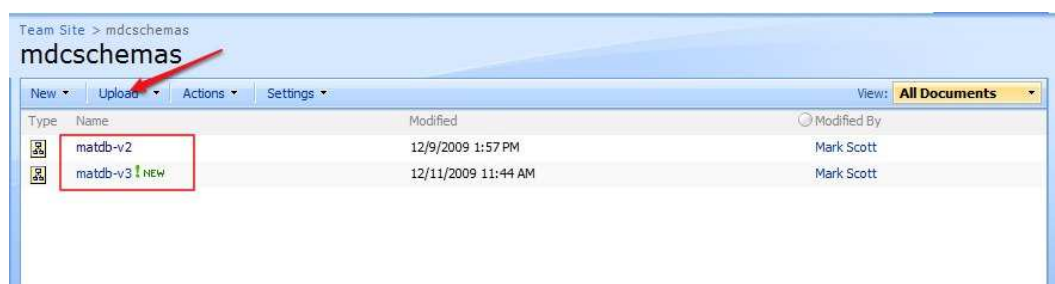
**GetURIForMDCID** `http://mdc/allitems/{MDCID}/uri`

This operation returns the URI of the requested object so it can be downloaded or displayed by EPrints. It is accessed via an HTTP GET request.

**Listing 3.1:** The header of a MatDB XML file defines the location of its schema – shown boldfaced – which is used for validation.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <MatDB:MatDB xsi:schemaLocation="http://odin.jrc.ec.europa.eu https://
  ↪ odin.jrc.ec.europa.eu/MatDB_XMLschema/matdb.xsd" xmlns:MatDB="http://
  ↪ odin.jrc.ec.europa.eu" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
  ↪ instance">...
```



**Figure 3.3:** XML schemas held in the mdcschemas document library that may be used for XML validation if they are not already publicly accessible elsewhere. The Upload button, indicated with the arrow, is used to add a schema to the document library.

## File validation

When XML data is uploaded it is validated against a schema. The schema to use is defined at the top of the XML file such as in Listing 3.1. As shown, the MatDB schema is stored at `https://odin.jrc.ec.europa.eu/MatDB_XMLschema/matdb.xsd`, but if this location was unreachable or an XML file with a different schema was to be used which was not available online, then one can be stored in the data centre in the mdcschemas document library and the XML file edited to use that location. Figure 3.3 shows two schemas stored in the mdcschemas document library, both custom versions of MatDB schemas used for testing.

The code in Listing 3.2 shows how the validation was performed. The XML validation routine is fairly simple: using the built-in Microsoft .NET Framework validation library it reads the XML file, throwing an exception if the file does not validate. This error is then reported to EPrints which can feed this back to the user.

## Data set upload

In order to upload files via the web service, it is required that a container is created for every publication. When the web service was designed, it was assumed that files would be uploaded first, and only associated with a container if required but it was later realised that this made the data store very disordered with some files in containers and others not. Therefore, the use of containers was made mandatory for all files. Because EPrints does not have a name for the container or have any knowledge of what containers already exist in the MDC, the container is created with a Globally Unique Identifier (GUID) (Microsoft 2014d) to guarantee uniqueness.



**Listing 3.2:** Code for validating the XML document.

```
1 // Set up the ability to resolve external XML resources
2 XmlUrlResolver xmlResolver = new XmlUrlResolver();
3 xmlResolver.Credentials = CredentialCache.DefaultCredentials;
4
5 // Set up the validation settings for an XmlReader
6 XmlReaderSettings xmlSettings = new XmlReaderSettings();
7 xmlSettings.XmlResolver = xmlResolver;
8 xmlSettings.ValidationType = ValidationType.Schema;
9 xmlSettings.ValidationFlags |= XmlSchemaValidationFlags.
  ↳ ProcessInlineSchema;
10 xmlSettings.ValidationFlags |= XmlSchemaValidationFlags.
  ↳ ProcessSchemaLocation;
11 xmlSettings.ValidationFlags |= XmlSchemaValidationFlags.
  ↳ ReportValidationWarnings;
12 xmlSettings.ConformanceLevel = ConformanceLevel.Document;
13
14 // Set the event handler in case of an error. The event handler, not
  ↳ listed here, throws an exception to indicate the file did not validate
  ↳ .
15 xmlSettings.ValidationEventHandler += new ValidationEventHandler(
  ↳ xmlSettings_ValidationEventHandler);
16
17 // Create the XmlReader using the settings we defined and read the entire
  ↳ document, throwing an exception if the validation fails at any point
18 XmlReader xmlReader = XmlReader.Create(fs, xmlSettings);
19 while (xmlReader.Read()) { }
```

After the EP2DC plug-in has created a container, files can be uploaded to it. When an XML file is uploaded, provided validation was successful if requested, it is stored in the document library as a file with the MatDBTest content type as we will show in Section 3.3.

### 3.2.3 EPrints plug-in

The EPrints plug-in itself is written in the Perl programming language, the language used for EPrints. The user is offered a choice of data centres into which to save their data and asked for metadata relating to the upload as shown in Figure 3.4. The plug-in stores the metadata inside EPrints along with the publication and its metadata, with the data files being transmitted to the data centre. The identifier returned by the data centre is also stored by EPrints for later use.

## 3.3 Results

The prototype system was developed to demonstrate key principles (rather than as a production system); to show the system operational, we present screenshots taken during testing. First, to test container creation, Figure 3.5 shows some containers created by EPrints as they appear in the SharePoint document library. The figure shows a number of folders with GUIDs for names where each folder is used to contain the data files making up a data set. Figure 3.6 shows a file in the document library deposited in one of the containers by EPrints; notice the

The screenshot shows a metadata entry form for a file named 'specimen\_1.xml' (7Kb). The form includes several sections with red star icons:

- Data Centre:** A dropdown menu showing 'Materials Data Centre', 'Neuropathology Data Centre', and 'AMC Chado Repository'.
- Test Type:** A dropdown menu showing 'Tensile', 'Creep', 'Fatigue', 'Impact', 'Fatigue Crack Growth', and 'Creep Crack Growth'.
- Test Date:** Fields for Year (2009), Month (November), and Day (26).
- Test Centre:** A text area containing 'University of Southampton'.
- Access control:** A dropdown menu showing 'Open access', 'Restricted to registered users', and 'On-demand'.

Buttons for 'Delete', 'Update Document', and 'Delete document' are visible. A link 'specimen\_1.xml 7Kb' and a question 'Need to a schema to this document?' are also present.

Figure 3.4: Metadata entry for a data file being deposited to an external data centre using the EP2DC plug-in.

Team Site > mdcdata

### mdcdata

New Upload Actions Settings View: All Documents

Type	Name	Modified	Modified By
Folder	0cbf35ca-c6ab-483d-a941-a24d2e4b285e	12/10/2009 4:45 PM	System Account
Folder	17e6ba26-3da1-4be3-a704-4f2d8399537d	12/10/2009 5:07 PM	System Account
Folder	20d2db2d-f3fa-4bbb-ae2f-f3f7314fb52e	12/10/2009 4:39 PM	System Account
Folder	25b5839f-2e6a-4dd8-a7a0-66129f0aabb	12/4/2009 9:10 PM	System Account
Folder	2654567b-fc65-4146-8e79-6b95b66c4b5e	12/7/2009 5:14 PM	System Account
Folder	28833ae7-42c4-4250-b38e-bc6f990ec87c	12/10/2009 5:02 PM	System Account
Folder	3329d0f9-1888-4eff-ab57-4da9cc071f64	12/10/2009 2:08 PM	System Account
Folder	358600f2-3e41-4937-bfec-56face202b43	12/10/2009 5:02 PM	System Account
Folder	3def0a05-743c-4a62-a089-0d54d60fa858	12/10/2009 6:01 PM	System Account
Folder	41c44b40-8670-43f2-a569-2e14fdd6a88c	12/10/2009 4:45 PM	System Account
Folder	4e680571-ea12-4d45-bac2-a3c3f71acee3	12/10/2009 5:01 PM	System Account
Folder	5b4dafd8-cba4-44f9-ad68-c62dcb957d32	12/10/2009 2:08 PM	System Account
Folder	614f8a80-94d9-4112-8768-ef3bea0f4fe0	12/10/2009 11:39 AM	System Account
Folder	637c646d-12a7-4008-97e2-2e677747b6f0	12/10/2009 4:40 PM	System Account

Figure 3.5: Containers are shown as folders in the MDC. GUIDs are used as the names of the folders.

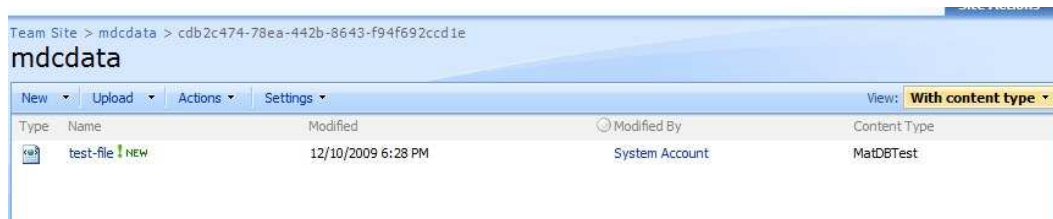


Figure 3.6: This file was deposited here by EPrints inside a container. This is how it is displayed in the MDC prototype's interface.

The screenshot shows the EP2DC EPrints interface. The top navigation bar includes links for Home, About, Browse by Year, Browse by Subject, Browse by Division, and Browse by Author. A search bar is present with a 'Search' button. The main content area displays the following information:

**Life assessment methods for industrial steam and gas turbines blade to disc interfaces**  
 Soady, K. and Reed, P.A.S. and Mellor, B.G. and Morris, A. (2009) *Life assessment methods for industrial steam and gas turbines blade to disc interfaces*. 15th Postgraduate Conference in Engineering Materials, Bio-engineering and Tribology . pp. 5-6. (Unpublished)

A Microsoft Word document icon is shown next to the title: [Microsoft Word \(Life assessment methods for industrial steam and gas turbines blade to disc interfaces\)](#) 3048Kb.

Below the document, a section titled 'EP2DC - data management and collation for EPrints' states: 'The following data is available to support this item:'.

Data available from: [Materials Data Centre](#)

Buttons for '+ Show details' and '- Hide related items' are present.

An XML icon is shown next to a list of files:

- [specimen\\_1\\_full.xml](#)
- [specimen\\_1\\_full.xml](#)
- [specimen\\_1\\_full.xml](#)
- [test-file.xml](#)
- [test-file.xml](#)

At the bottom, a table provides metadata for the item:

Item Type:	Article
Subjects:	<a href="#">T Technology &gt; TA Engineering (General)</a> , <a href="#">Civil engineering (General)</a>
Divisions:	<a href="#">Faculty of Engineering, Science and Mathematics &gt; School of Engineering Sciences</a>
ID Code:	30
Deposited By:	Mr Mark Scott
Deposited On:	30 Mar 2010 14:29
Last Modified:	30 Mar 2010 14:29

Repository Staff Only: [item control page](#)

At the bottom, a footer states: 'EP2DC is powered by [EPrints 3](#) which is developed by the [School of Electronics and Computer Science](#) at the University of Southampton. [More information and software credits.](#)' The EPrints logo is also present.

Figure 3.7: How EPrints displays the publication and the information given to it by the MDC.

content type is displayed as `MatDBTest`. Finally, Figure 3.7 shows a test EPrints deposit as it is rendered to the user. The figure shows the related items returned by the data centre and integrated into the EPrints page when viewing a deposited publication with an associated data set. The related items are all files with the `.xml` extension; in fact, these files are actually all files with the `MatDBTest` content type with other files in the document library being ignored.

### 3.4 Summary

In this chapter, we showed the EP2DC project and was able to demonstrate the concept of storing data in a remote data centre whilst publishing a paper, proving that EPrints could be used as a front-end to a data centre. It provided the ability for the data centre to provide a list of related data items back to the user in their EPrints interface based on the type of file, and also permitted the storage and validation of MatDB data files, improving data integrity.

There were a number of inadequacies with this model. In order to achieve our goals for the Materials Data Centre, further work to enhance EP2DC includes:

- Enable named containers, rather than using GUIDs. For a user-driven data management system, using GUIDs for folder names would be unacceptable.
- Although the web services provided by the Materials Data Centre service layer are fully REST compliant, there are existing protocols for communicating with repositories, discussed in Section 2.10, which may provide better acceptance.
- EPrints should pass metadata it has about the publication to the data centre so the publication can be found from the data.
- It is insufficient to just support MatDB XML data for materials engineering as was shown in Section 2.5.3. The data centre will need to be improved to support other file types.
- The operations provided do not permit a way of specifying the data type being deposited.

The use of SharePoint provided a good starting point for the heterogeneous data requirements of data centres such as the Materials Data Centre, although the inability to store files larger than 2 GB, the necessity of data sets with names as GUIDs to guarantee uniqueness and the requirement that uploads must be performed using a web service is not satisfactory because it limits the use of the data centre and does not permit storage of all types of data discussed in Section 2.6.

In the next chapter, we will show a more advanced model for storing a variety of data which was developed to support larger data files, data sets with user-specifiable names and upload without requiring the use of a web service.



## Chapter 4

# A model for managing materials data


---

The work described in this chapter has been published as:

Mark Scott, Richard P. Boardman, Philippa A. Reed, Tim Austin, Steven J. Johnston, Kenji Takeda and Simon J. Cox. 2014. 'A framework for user driven data management'. *Information Systems* 42 (June): 36–58. doi:10.1016/j.is.2013.11.004

---

### 4.1 Introduction

N THIS CHAPTER we introduce a model that was created for the Materials Data Centre (MDC) project for the management of the wide variety of data produced by the materials engineers, including that discussed in Section 2.6. It uses a novel combination of a flexible, user specifiable and compoundable metadata database connected to a generic data store, in this case a file system. The architecture permits: the storage and sharing of any data files, limited in size only by the file system; the ability for users to define their own metadata and metadata structures; and relationships between data sets to be defined. It improves on the early MDC prototype shown in Chapter 3 for the EP2DC project, removing restrictions of a 2 GB limit on file size and the requirement to use GUIDs as the name of the data set, and improves its metadata support. The features of the system include:

- Storage and sharing of any data files, limited in size only by the file system.
- The addition of metadata to a data set, with the ability for users to define their own data structures, with nesting.
- The support of relationships between data sets.
- Allowing predefined metadata to be created as a template and for metadata to be copied between data sets.
- Encouraging use of the system by providing data set recommendations.
- Allowing plug-ins, in order for the system to provide customised reports and tools depending on the data.
- Supporting the following data types, introduced in Section 2.6:
  - Tensile test

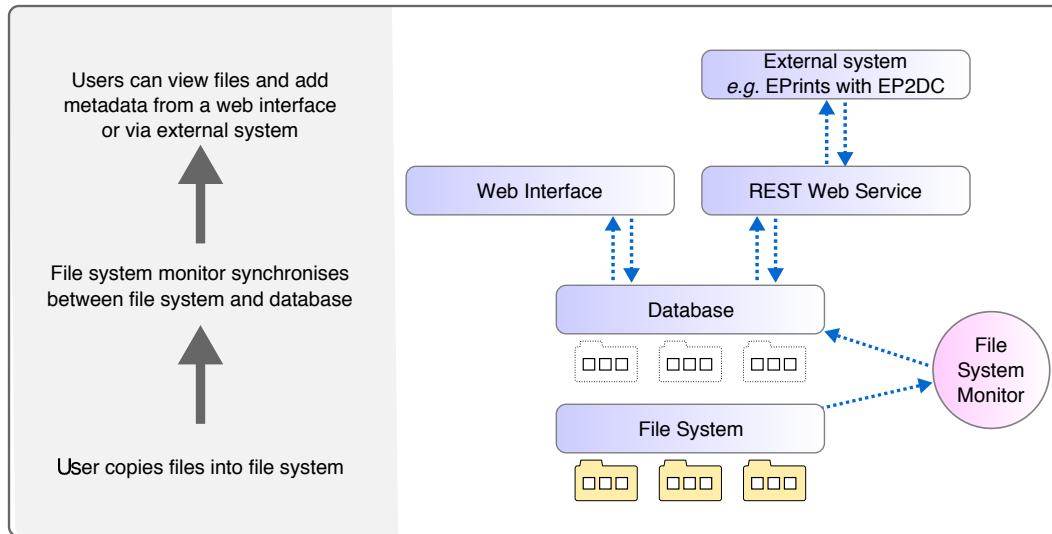


Figure 4.1: Data flow in data management model.

- Fatigue test
- Fractography
- Microfocus computed tomography data

We will first discuss the architecture of the model, including the technologies used in our implementation, then we show some of the key interface features added to assist in the capture and management of data and metadata.

## 4.2 Architecture

The final architecture of our data management model consists of four components: a file system which permits users to store files; a metadata database that permit users to define their own data structures; a file system monitoring service that synchronises changes in the file system with the database; and an interface to manage the data sets, their metadata and their relationships to other data sets. Figure 4.1 illustrates this, showing how the user can interact with the metadata database using the web interface once files have been copied into the file system. The web service built for the EP2DC project, and discussed in the previous chapter, provides another entry point which we will show in Chapter 5.

Our implementation of the metadata database is hosted on a Microsoft SQL Server 2008 R2 (SP2) Standard Edition cluster managed by the University of Southampton's central IT service and was tested on SQL Server 2008 SP1. The file monitoring service runs on a Microsoft Windows Server 2008 R2 virtual machine with 4 GB of RAM using the NTFS file system (Ruslinovich 2000) for data storage accessed by users via a Samba share. The Microsoft SharePoint site and accompanying database is hosted on the same machine. The virtual machine itself is hosted using VMware ESX server by the central IT service.

The technology choices of NTFS for the file system, Microsoft .NET Framework for the file system monitor and interface tools, and Microsoft SQL Server for the database evolved

from the choice of Microsoft SharePoint for the EP2DC project in Section 3.2.1. While the decision to stop storing data files within SharePoint was made to work around the 2 GB limit of SharePoint, we decided to keep Microsoft SharePoint for the interface for the same reasons as discussed in Section 3.2.1. Interface components (discussed in Section 4.2.5) and an API (discussed in Section 4.2.3) were then created with the Microsoft .NET Framework, as supported by Microsoft SharePoint. Building the file system monitor with the same API and hosting it on a Microsoft Windows file server was then a natural choice which led to the use of NTFS as the file system discussed in Section 4.2.1. The metadata, discussed in Section 4.2.2, is held in Microsoft SQL Server; this was chosen as it was a good fit for working with Microsoft SharePoint and the other Microsoft technologies, and was fully supported at our institution, but no specific features were used that would prevent it from being hosted in another database product.

### 4.2.1 File system

Choosing a file store as the base of the design allows us to support the wide ranging data types required, ensures flexibility and reliability, and allows the system to seamlessly take advantage of developments in file system technologies and benefit from more advanced file systems such as a distributed file system. It also removes any limits imposed by some systems that rely on a database for the storage of their data and simplifies the deposit process whilst keeping backups of data straightforward.

Making the entry point to the system a file share allows other tools that have no knowledge of the data centre to write data sets to the MDC; CT data shown in Section 4.3.4 could be deposited directly by the scanner. Conversely, by using the metadata in the database, we can identify data sets that can then be read back via the file system which we will demonstrate in Section 4.6.5.

The expected performance impact of choosing a file system over a database-based system depends on the type of data stored: the accepted rule of thumb is that a file system is more efficient with larger files whilst databases are best suited to large amounts of small objects and will outperform a file system in this case. Sears, Ingen and Gray (2006), comparing Microsoft SQL Server 2005 (Microsoft 2014f) to the NTFS file system, showed that 1 MiB is the maximum size of file that can be stored in a database without hindering performance. For anything below 256 KiB in size, the database was most suitable and for file sizes between 256 KiB and 1 MiB, the results varied depending on the workload.

The choice of file system could greatly affect the flexibility of the system as indicated in Section 2.4.2. The file size supported by the file system is particularly important when choosing a file system to support a system such as this and, by changing the file system, it is possible to introduce additional features such as file replication without affecting the capabilities of the file system monitor. In our implementation of this model, the NTFS file system supports file sizes of up to nearly 16 TiB ( $2^{44}$  B minus 64 KiB) under Microsoft Windows 2008 (Microsoft 2014b, 2003).



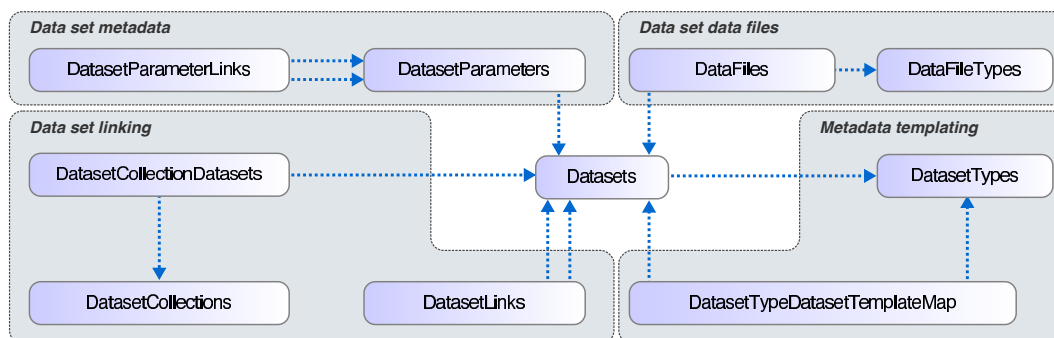


Figure 4.2: Metadata database tables in data management model.

## 4.2.2 Metadata database

The metadata database keeps track of data set and file metadata, and the data set relationships that we discussed in Section 2.6.7. The database model shown in Figure 4.2 holds five types of data: data sets, data files, data set metadata, data set relationships, and metadata templates.

The data set is the principal data construct in the system and is used to collect related data files and metadata. To give an example from materials engineering, a data set could be a collection of files relating to an experiment such as a fatigue test and associated metadata such as the last time the equipment was calibrated. During the course of the project, we have also used the terms ‘container’ and ‘experiment’ to mean the same thing and its analogue in the file system is a folder.

The database and file system are kept synchronised by a file system monitor which will be described in Section 4.2.4. When folders are created on disk, a data set record representing the folder is created in the database, and as files are added to the folders, they are represented in the database as data file records tied to each data set. Data sets can be tagged with metadata, and relationships between data sets can be defined. Templates are used for holding predefined metadata which can then be used for populating a data set’s metadata.

Table 4.1 gives a summary of the tables in the database; we will present the final schema in Chapter 5 (see Figure 5.1). The `Datasets` table holds the name of the folder and an optional description, and information about the files in the folder is recorded in the `DataFiles` table. It is possible to type each data set and data file with the `DatasetTypes` and `DataFileTypes` tables. This can help users to categorise their data and will be useful in queries and providing supporting data to interface tools. The data set typing capability has been used to help with data set discovery (see the discussion about data set recommendations in Section 4.2.3) and with features such as metadata templates described below.

The remaining tables are used to manage the data set metadata, metadata templating and data set relationship features, and are discussed in the following sections.

### Hierarchical data set metadata

The `DatasetParameters` table allows data set metadata to be recorded using an extended EAV model. As with the standard EAV model, there is an entity, which is the data set the parameter is linked to, an attribute, which is the parameter’s name and a parameter value.

**Table 4.1:** Descriptions of the metadata database tables.

Database Table	Description
Datasets	A collection of data files and metadata.
DataFiles	A list of the data files known to the system.
DatasetParameters	Each parameter can contain Name, Value, Unit and Type fields for tagging a data set with metadata. The IsCompulsory field controls whether a value is required in the parameter's value.
DatasetParameterLinks	Permits nesting of metadata.
DatasetTypes	The types of data sets known to the system.
DataFileTypes	The types of the data files known to the system.
DatasetTypeDatasetTemplateMap	A list of data sets that can be used as templates for data sets of a specified type.
DatasetLinks	Contains the ID of two data sets providing a direct link between them.
DatasetCollections	Collections of data sets to group containers together.
DatasetCollectionDatasets	The data sets in a collection.

**Table 4.2:** Fields in the DatasetParameters table used to describe metadata entries.

Field	EAV	Description
DatasetID	Entity	ID of the data set the metadata is linked to
Name	Attribute	Attribute name
Value	Value	Attribute value
Unit	Extension	Unit for the attribute value
Type	Extension	Data type for the value, for application layer validation
IsCompulsory	Extension	As the Value field is nullable to permit optional parameters, this boolean field gives the ability for error checking
SourceParameterID	n/a	Optional parameter ID. If the parameter was copied from a template, this field records the original parameter's ID

In addition to the EAV model, there are three other fields: Unit, Type and IsCompulsory. Unit permits the ability to record the unit if the parameter was a measurement, and Type and IsCompulsory permit application layer validation to ensure that required values were not omitted and were of the correct format. The fields are summarised in Table 4.2.

In order for users to create more complex data structures, the model is extended further with the addition of the DatasetParameterLinks table to permit nesting of parameters. This enables powerful tagging capabilities.

### Data set metadata templates

Templates permit parameters for different data set types to be predefined. These parameters are then copied into the data set's list of parameters and, because the parameter copy is linked to the original, changes to the original or copy can be identified when necessary.

Templates are stored in the Datasets table because they follow the same structure as an ordinary data set. Template data sets have a special type that defines them in the system as a template, but copying of parameters from a template to a data set, from data set to data set or from template to template is possible because the system considers them equivalent.

The DatasetTypeDatasetTemplateMap table, in combination with the DatasetTypes table, links templates to types of data set.

## **Data set relationships**

There are three ways of linking data sets. The first is through a direct link in the Dataset-Links table which associates one data set with another. The second option is to create a new collection in the DatasetCollections table and add a list of related data sets to the collection, stored in the DatasetCollectionDatasets table. Finally, at the user interface level, keywords inside parameter values can be used to render links to other data sets. For example, using the string 'Dataset 8' inside a parameter value creates a link to the data set with the ID of 8.

### **4.2.3 Logic layer**

In order to streamline interaction with the database, such as simplifying the management of nested metadata, we provided a Microsoft .NET Entity Framework database model along with a number of extension functions to help to build the software supporting the system. This essentially provides an API (Application Programming Interface) to enable more powerful features to be built as they are required.

The features of the API will be discussed in this section. Much of this feature set is already exposed in the interface; components built upon this will be discussed in Section 4.2.5 and examples of using the API will be shown in Section 4.6.5.

## **Metadata manipulation**

Software code that fully takes advantage of the nesting of metadata in the database can prove complicated, so a reliable set of functions for manipulating these is necessary. We provide the following capabilities in the data access library:

- **Retrieving parameters, with or without nesting.**
- **Retrieving an individual parameter's parents or children.**
- **Adding or updating metadata, and creating hierarchical metadata** – e.g. deleting a parameter tree.
- **Copying metadata between data sets** – To make metadata entry easier, parameters can be copied between data sets, either from another data set or from a template linked to a data set's type. This is crucial in order to support the template feature, and make data entry for users easier.
- **Parameter validation** – Such as checking the Type and IsCompulsory fields.
- **System parameter related functions** – For creating and retrieving parameters under a hidden parent in a data set for system parameters such as the data set's thumbnail.

## **Relationship management**

Data sets can be related by links and collections. Functions are provided to add and delete these, as well as maintain the link between the data set type and data set templates.

## Searching

A number of search functions are provided. The simplest involve finding data sets and data files by their IDs. It is also possible to retrieve data files using the parent data set's ID and data sets using a collection ID. More powerful functions allow searching for data sets by name and their parameters. Search functions provided are:

- Finding data sets by name (start of string)
- Finding data sets by parameters (by start of name; by exact name and optionally value)
- Finding data set by substring (searches for data sets by data set name, data file name and parameter name)

Interface controls that use these search functions are discussed in Section 4.2.5.

**Data set recommendations:** As discussed in Section 4.2.2, relationships can be created between data sets which helps users to manage their data sets. To increase awareness of other data sets, the linking and typing information is used by the system to recommend related work to the user. It does this by looking at data sets that have been linked to, and finding what other data sets also link to those data sets. Using a materials engineering example, if an engineer links an experiment to a material information sheet, the system suggests other experiments that also used that material. It can also suggest data sets that are of the same type (such as other fatigue tests).

### 4.2.4 File system monitor

With files stored in the file system on a server, and metadata about those files in a database, we introduced a synchronisation service to update the database, ensuring that users can see their files in the interface almost immediately, that associated data set and file metadata can be kept even after the original files have been deleted and, even if the file share is not available, associated metadata can still be viewed and edited.

The file system monitor recognises when new data sets and data files have been added to the file share and records them in the database so metadata can be added by the user. Using the file's time stamps, SHA-1 hash, name and path, the service can identify file renames, additions, deletions, copies and renames. A scheduled task ensures that missed file system events (when the monitoring service is not running or buffer sizes being exceeded when large amounts of changes are made (Microsoft 2010a)) are also picked up to ensure metadata remains attached to the correct data sets and files. Locking and contention (when the file is in use or still being updated) are tackled through exception handling in the file system monitor so this is minimally intrusive to the user.

This section provides more details on the internal operation of the file system monitor. Listing 4.1 shows the file system monitor processing items on its queue.

**Listing 4.1:** Excerpt from the file system monitoring service log showing the monitoring of data sets, finding a file deletion (line 1), file addition (line 2), data set creation (line 3), data set rename (line 4), trigger of consistency check (line 12), file rename (line 16) and a data set deletion (line 20). The service will also check for changed files within a data set and, with the improvements made in Chapter 5, will modify the security of folders based on the settings made by a data set owner using the interface.

```

1 15/02/2014 15:04:58: Deleted F:\data\ta10g11\DMS4M\DSM4M.png
2 15/02/2014 15:05:26: Adding file DSM4M.png to experiment 130 (DMS4M)
3 16/02/2014 14:15:11: Creating database record for experiment New folder
4 16/02/2014 14:15:19: Folder 'DSM4M Creep Test' has moved: F:\data\ta10g11\New
  folder => F:\data\ta10g11\DSM4M Creep Test. Updating metadata
5 16/02/2014 14:15:29: Folder 'DSM4M Creep Test 870C 450MPa' has moved: F:\data\
  ta10g11\DSM4M Creep Test => F:\data\ta10g11\DSM4M Creep Test
  870C 450MPa. Updating metadata
6 16/02/2014 14:15:39: Adding file DSM4M 870C 450MPa to experiment 131 (DSM4M Creep
  Test 870C 450MPa)
7 16/02/2014 14:38:56: Creating database record for experiment New folder
8 16/02/2014 14:39:04: Folder 'DSM4M 982C 240MPa' has moved: F:\data\ta10g11\New
  folder => F:\data\ta10g11\DSM4M 982C 240MPa. Updating
  metadata
9 16/02/2014 14:39:14: Adding file DSM4M 982C 240MPa to experiment 132 (DSM4M 982C
  240MPa)
10 16/02/2014 14:47:16: Creating database record for experiment New folder
11 16/02/2014 14:47:19: Folder 'DSM4M Creep Test' has moved: F:\data\ta10g11\New
  folder => F:\data\ta10g11\DSM4M Creep Test. Updating metadata
12 16/02/2014 14:47:31: Folder F:\data\ta10g11\DSM4M Creep Test does not exist,
  ignoring it. Triggering folder checking to clear up deleted
  folders.
13 16/02/2014 14:47:34: Folder 'DSM4M Creep Test 982C 320MPa' has moved: F:\data\
  ta10g11\DSM4M Creep Test => F:\data\ta10g11\DSM4M Creep Test
  982C 320MPa. Updating metadata
14 16/02/2014 14:47:44: Adding file DSM4M 982C 320MPa to experiment 133 (DSM4M Creep
  Test 982C 320MPa)
15 16/02/2014 15:00:24: Folder 'DSM4M Creep Test 982C 240MPa' has moved: F:\data\
  ta10g11\DSM4M 982C 240MPa => F:\data\ta10g11\DSM4M Creep Test
  982C 240MPa. Updating metadata
16 16/02/2014 15:00:24: File moved F:\data\ta10g11\DSM4M 982C 240MPa\DSM4M 982C 240
  MPa => F:\data\ta10g11\DSM4M Creep Test 982C 240MPa\DSM4M 982
  C 240MPa
17 16/02/2014 15:05:26: Creating database record for experiment New folder
18 16/02/2014 15:05:31: Folder F:\data\ta10g11\New folder does not exist, ignoring it
  . Triggering folder checking to clear up deleted folders.
19 16/02/2014 15:05:34: Folder 'DZ951' has moved: F:\data\ta10g11\New folder => F:\
  data\ta10g11\DZ951. Updating metadata
20 16/02/2014 15:08:34: Deleted from database 134 F:\data\ta10g11\DZ951

```

## Tracking data sets

The file system monitor communicates with the database to keep folder and data file names synchronised. It reads the data set's ID from a tracking file ('metadata.xml' by default) stored in a hidden directory in each folder ('.dataset' by default) when starting up to identify the exact record to read from the database, in preference to relying on the name of the directory in the file system in case it was renamed while offline. This permits the following situations to be handled:

- **Folder created.** If a folder is identified without a tracking file, a new '.dataset' folder, database record and a tracking file containing the database record's ID are created.

- **Folder renamed or moved.** The tracking file's recorded data set ID is used to look up the name in the database and compare it with the actual folder name. The name in the database is updated.
- **Copied folder.** If we find that the name in the database is different, and in addition the old folder still exists and contains a tracking file with the same ID, the new folder is considered to be a copy. In this situation, the tracking file in the copied folder is renamed and a new database record and tracking file is created with a new ID.

### Tracking file changes

On a file system event, we trigger a check of just the data set that changed. Rather than check only the file that was changed, we check the entire data set. This ensures we do not miss, for example, file renames if the event generated does not reflect exactly what happened, or if other events were missed. Reducing the time taken to perform this full recheck is discussed later in the section.

The file system monitor starts with a list of all files in the file system and a list of the files recorded in the database, then follows a procedure to categorise the files as updated, renamed and unchanged, eventually leaving a list of files that are considered new and deleted. The order this is done is particularly important to ensure the correct change is identified. The steps used are as follows:

1. **Obtain a list of files on the file system.** Metadata collected about files include: last access time, last write time, creation time and SHA-1 hash.
2. **Obtain a list of files recorded in the database.**
3. **Identify files with updated metadata and remove from both lists.** Unchanged properties: file name, file size and SHA-1 hash. Changed properties: timestamps.
4. **Identify unchanged files and remove from both lists.** Unchanged properties: file name, file size and SHA-1 hash.
5. **Identify renamed files and remove from both lists.** Changed properties: file name.
6. **Identify updated files and remove from both lists.** Unchanged properties: file name. Changed properties: file size, SHA-1 hash, or time stamps.
7. **Files remaining in database list are deleted.** Any files still in the database that do not match the above criteria should be considered deleted.
8. **Files remaining in file system list are new.** Any files on the file system that have not been found in the database using the criteria above are considered new.
9. **Synchronise changes.** Process the lists of new, updated, renamed and deleted files by modifying the database.

## Ensuring consistency

To ensure the database remains consistent in the case of missed events, we schedule the following checks at regular intervals:

**File system checks:** We trigger a check of each data set on disk for comparison against the database. The check performed is the same as when an event occurs in a data set on the file system, generated artificially for every folder on disk.

**Database metadata checks:** We check all the files and data sets in the database against what is actually on the file system. This allows us to identify missing data sets and data files, and mark them as deleted. This check is also sometimes triggered when dealing with a file system event if the folder that generated an event is found to be missing.

Again, the order in which this check performs its steps is important. A list of data sets in the database and a dictionary of data set tracking files in the watch folder, keyed by the data set's ID, is generated and then the following steps are performed for each data set in the database:

1. **Find the matching tracking file in the dictionary using the ID from the database record.**
2. **If tracking file has not been found, try to load it from the location saved in the database.** It is possible that the database has the correct information if it has been set manually or the watch folder has been changed.
3. **If the tracking file's data set ID matches the database's ID, compare the data set at the path found with the database.** This is to identify renames and moves, triggering a complete folder check if a change is found.
4. **Any database records where a matching tracking file could not be found are deleted.** Mark data sets that could not be found on disk as deleted in the database.

## Tracking changes while offline

The file monitoring service may not always be active and file modifications may be missed. Therefore, on start up, all files and data sets are rechecked using the same criteria as above to detect any changes that were missed.

## Reducing the time for checking large data sets

The generation of a SHA-1 hash can be expensive for large data sets such as a CT scan. As we rescan the entire data set every time any file changes, and we regularly recheck all files in all data sets, we only regenerate the SHA-1 hash if any of following criteria have changed: File name; Last write time; Creation time; SHA-1 hash generated time is earlier than last write time; and File size. If all of the criteria match we use the SHA-1 hash stored in the database. This drastically improves the speed of checking a large data set.

☐ Show deleted datasets



ID	Dataset name	Dataset info	Dataset type	
5	Material N18	N18 Datasheet 1 data files 17 parameters	Material data sheet	 Thumbnail
6	Material FV448	FV448 Datasheet 1 data files 17 parameters	Material data sheet	 Thumbnail
7	N18 Fatigue Test	Fatigue Test of N18 1 data files 0 parameters	Fatigue test	
8	N18 Failure CT Scan	CT Scan of N18 Failure 1 data files 0 parameters	CT scan	
9	Materials data sheet template	Material data sheet example parameters 0 data files 17 parameters	Template	
10	Dublin core template	Dublin core parameters 1 data files 15 parameters	Template	

Figure 4.3: Screenshot showing list of a user's data sets, including a thumbnail if present and data set summary.

## 4.2.5 Interface

### Data set deposit

Data sets are deposited by the user by creating a folder in a network share. The file system monitor component creates a stub in the metadata database corresponding to the folder that has been created.

A list of the user's deposited data sets is available using the web interface, including ones that have been deleted that might still contain metadata of interest, and shows the number of data files and parameters in each data set, along with a description and optional thumbnail (see Figure 4.3). A web page for each data set allows viewing of data files in the data set, its metadata and relationships.

### Metadata and metadata tools

Once a data set has been created, metadata can be added. This includes providing descriptions for the data set and data files, setting the data set's type, and supplying other relevant metadata. The metadata can be entered manually by the user through the data set's web page, or added automatically by a plug-in. The metadata related features we provided are discussed in this section.

**Data set metadata:** The data set can be given a description and a type (see Figure 4.4(1)). While this helps the user to manage their data, the system makes use of the description in reports and the type is useful for suggesting metadata templates. Likewise, the data files in the data set can also be given a description which is used as a caption for images in reports.

A data set can be further described by adding metadata. This could include the conditions of an experiment or further information about a file in the data set. Each metadata entry can contain Name, Value, Unit, Type fields, and an IsCompulsory boolean field is particularly useful for setting up templates for metadata. The entries can be nested, further enhancing the complexity of the metadata (see Figure 4.5(a)).

These features allow us to support many metadata standards, depending on the data set owner's requirements.



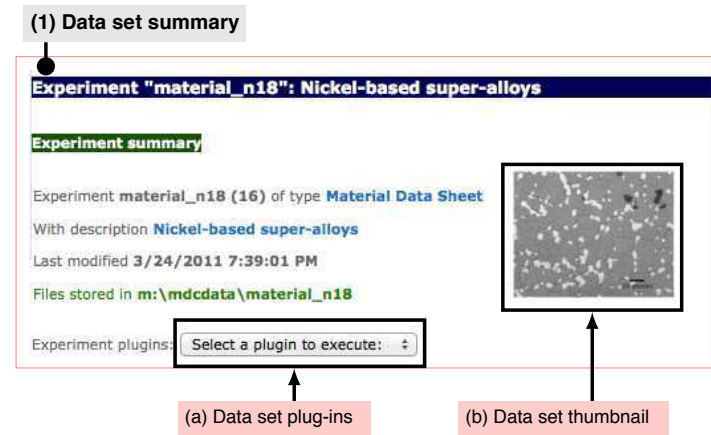


Figure 4.4: Data set details page – (1) summary section, showing (a) data set plug-ins and (b) a data set thumbnail.

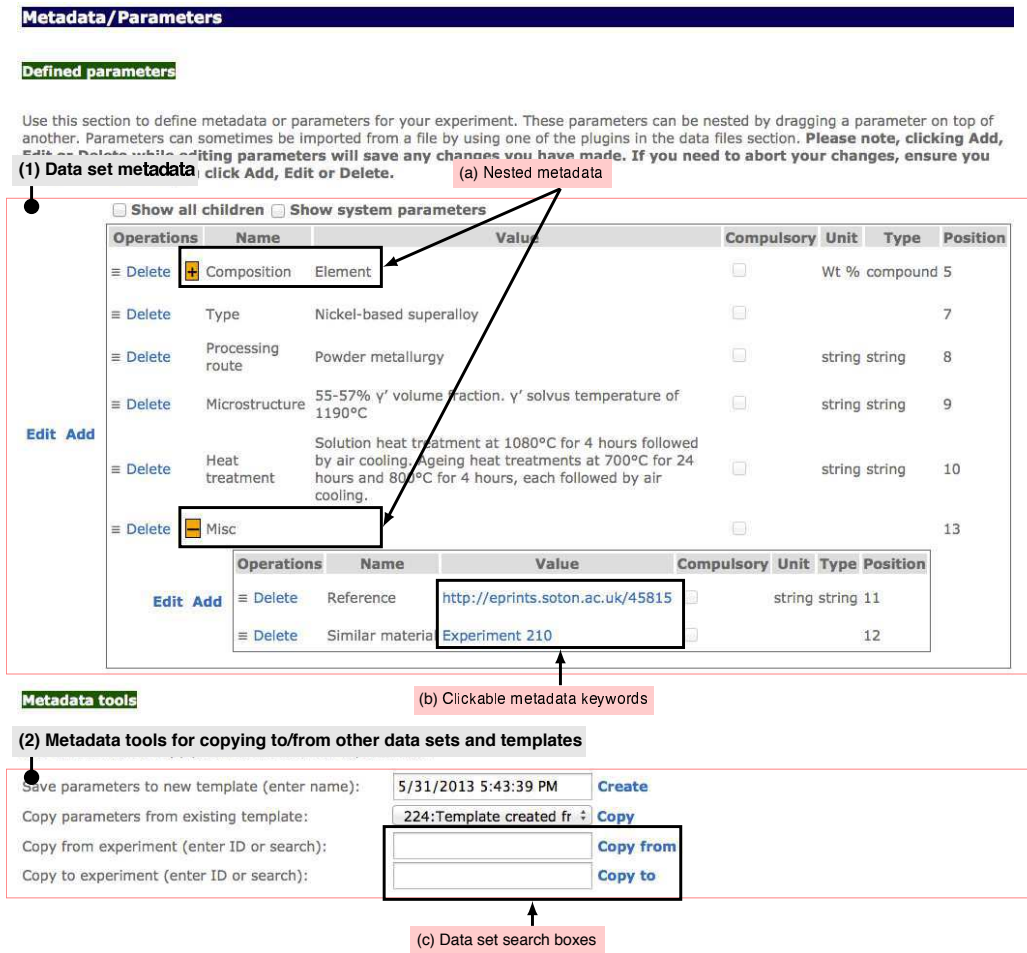


Figure 4.5: Data set details page – metadata sections: (1) metadata and (2) metadata tools. Shows (a) nested metadata, (b) clickable metadata keywords and (c) data set search boxes.

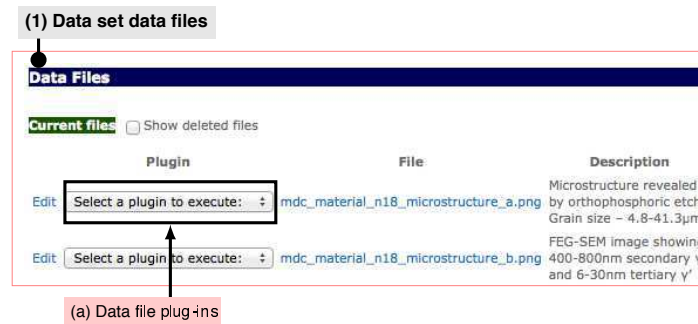


Figure 4.6: Data set details page – (1) data files section, showing (a) data file plug-ins.

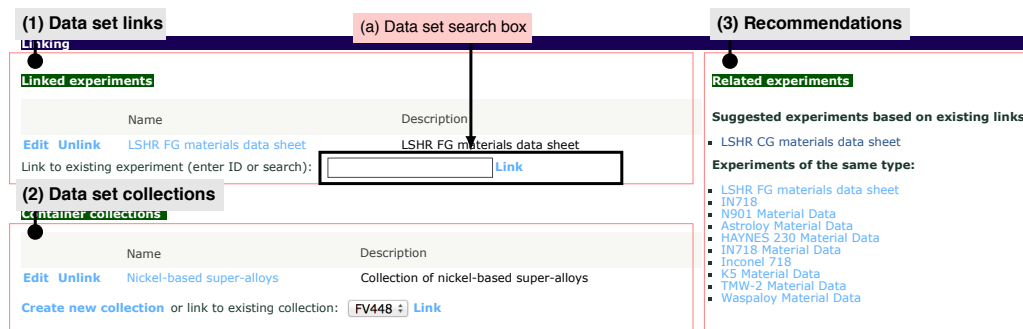


Figure 4.7: Data set details page – relationship sections: (1) data set links, (2) data set collections and (3) data set recommendations. Shows (a) a data set search box.

**Data set metadata copying and templates:** For users with more complex metadata – which would be repetitive and time-consuming to add every time – parameter templating features were provided, and interface tools were added for copying metadata to and from other data sets, or to and from a metadata template. Templates are linked to data set types, so a user who has set the data set's type correctly will have a choice of parameter templates that have been defined for that data set's type.

Users preferred for the system to allow incomplete or improper metadata – particularly with templates containing large amounts of metadata – allowing metadata important to the user to be completed quickly for use in reports or other tools; therefore any erroneous metadata is accepted but is highlighted to ensure the user knows they have missed a compulsory value or provided text instead of a number. In testing of the system, users felt strongly about having this flexibility (Reed and Soady 2010; Austin and Scott 2014).

**Data file metadata:** Users can view a list of data files in the data set from the data set details page and also provide descriptions for data files as shown in Figure 4.6. The data file description can also be used by the system in reports, such as for the caption for an image.

### Data set relationships

Data sets can be managed further by defining their relationships with other data sets. This can be done in three ways:

1. Direct link to another data set (see Figure 4.7(1)).

2. By creating a collection and adding the data set to that collection – this is analogous to tagging features found in some systems (see Figure 4.7(2)).
3. By referring to a data set using keywords in a metadata entry, such as ‘Experiment 27’ or ‘Dataset 27’. This becomes clickable when the metadata entry is rendered (see Figure 4.5(b)).

These relationships are used by users to manage their data, but they are also used by the system to recommend other data sets that might be of interest as discussed in Section 4.2.3.

### **Data set search**

Simple searching is possible using the Full Search (Figure 4.8) and Quick Search (Figure 4.9) controls.

The Full Search control takes a search term and returns three sets of results: data sets with names that match the term; data sets with parameters that match the term; and a combined list of data sets with names, parameters and files matching the term.

The Quick Search control is used on pages where space is limited but the user needs to find a data set quickly, such as when trying to copy metadata from another data set. It performs the combined search, listing data sets that match by name, metadata contents and file-name.

### **Plug-ins**

There are two types of plug-in. The first is a data set-level plug-in such as a report and the second is a file-level plug-in, such as the CT data browser shown in Figure 4.10 or a metadata import plug-in. File-level plug-ins are activated by selecting them from a dropdown list next to each file’s name (see Figure 4.6(a)).

## **4.3 Materials engineering use cases**

Section 2.5.1 discussed five potential use cases chosen from materials engineering for the Materials Data Centre: tensile testing, long crack threshold fatigue testing, plain bend bar testing for S-N curves, plain bend bar testing for short crack initiation and growth with replication, and fractography. Additionally, Section 2.6.5 discussed microfocus computed tomography and Section 2.6 showed material information used for describing a material. This section demonstrates the use of the system with a tensile test, fatigue test, fractography data, computed tomography data and a material information data set. The examples were created with help from the users of the system; the materials information and fatigue test use cases have been produced with data from Jiang et al. (2013) and the tensile test data is from Gabb et al. (2013).

### **4.3.1 Material information**

The material information use case’s aim is to record metadata about the material used in tests and makes good use of the metadata features of the system as well as accompanying data files.

Experiment search:

Experiment name results:  
Materials Data Sheet Template

Param results:

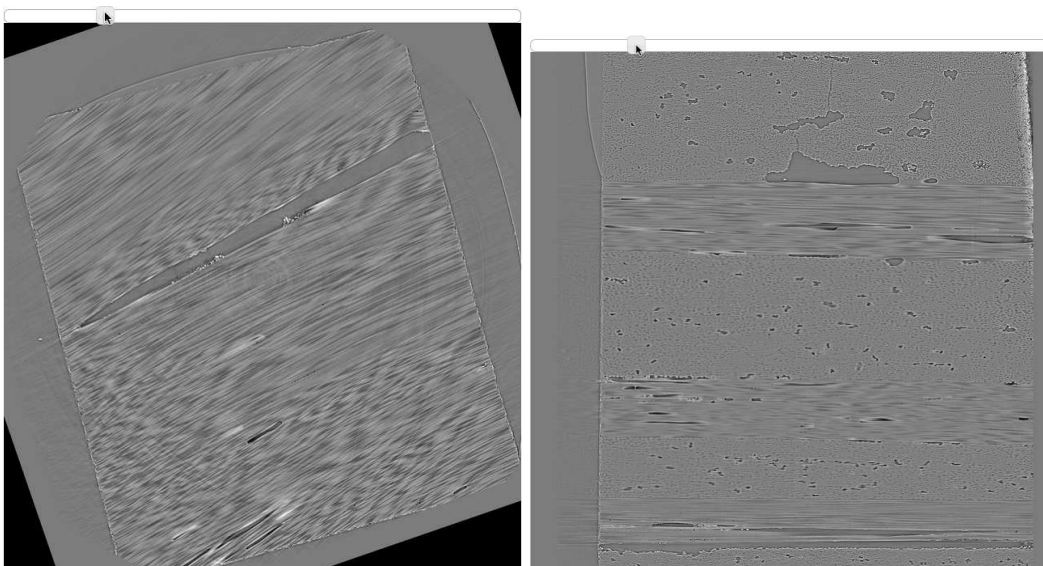
All results:  
Experiment:LSHR CG materials data sheet  
Experiment:Materials Data Sheet Template  
Experiment:LSHR FG materials data sheet  
Data file:materials\_windaq.png

**Figure 4.8:** Full search interface component for performing searches to find data sets. There are three boxes for reporting the search results: the first box shows data set matches found by data set name, the second box shows data sets that have a matching metadata parameter, and the third box combines results from the data set name, metadata and file name searches into a single results list.

Experiment quick search:

Experiment:LSHR CG materials data sheet  
Experiment:Materials Data Sheet Template  
Experiment:LSHR FG materials data sheet  
Data file:materials\_windaq.png

**Figure 4.9:** Quick search interface component for performing searches by data set name, metadata parameter and file name. It is used on web pages as a convenient way to find data sets for linking or copying metadata.



**Figure 4.10:** Slices from the x-y and y-z planes extracted from a CT scan of a material using the image browser plug-in.

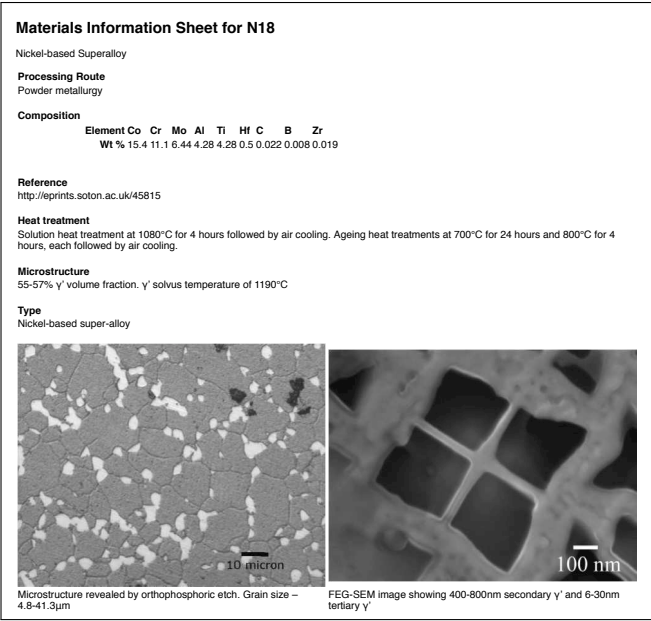


Figure 4.11: MDC reporting example: a material information sheet showing a material’s composition (generated from the data set’s metadata) and its microstructure (using files in the data set).

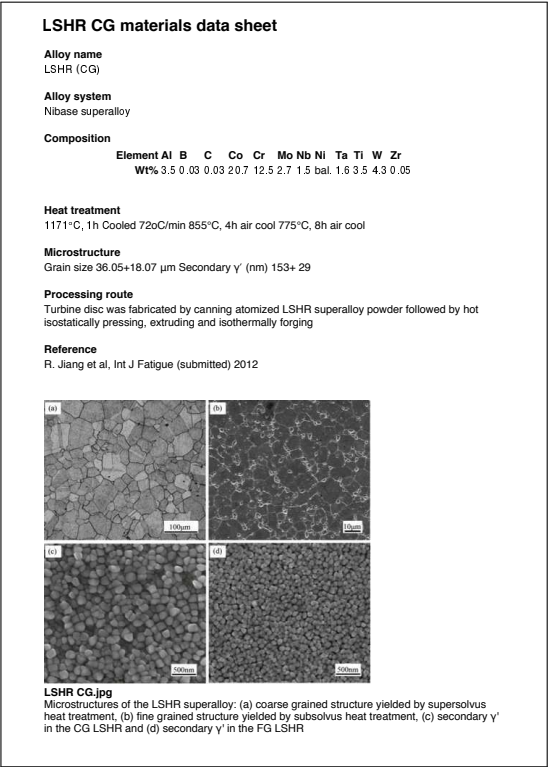


Figure 4.12: Material information sheet produced using the data set reporting plug-in.

Engineers at the University of Southampton produce materials data sheets, discussed in Section 2.6.6, containing composition, heat treatment and microstructure data which can be referred to as necessary. The data collected about a material was also discussed in Section 2.6.6. To replicate this capability with the MDC, the user would create a folder in the file share and copy any material microstructure images. Information about a material can then be entered using the interface in name-value format; a template containing common material properties can be used to assist with this. The example material information data, given in Figure 2.3 in Section 2.6.6, was entered into the MDC and the report is shown in Figure 4.11 which is automatically formatted and looks similar to the existing report.

The report produced for the LSHR CG material is shown in Figure 4.12; the images and descriptions are included automatically, and any nested parameters are drawn as a table.

To obtain the best reports, users need to provide a description for the data set and images, or name their files appropriately (if no description is given, the file name is used as a caption). The correct templates can only be chosen if the user changes the data set type to a material data sheet. Descriptions and types are part of the Dublin Core metadata set so this is useful metadata to capture for the repository.

#### 4.3.2 Tensile test

The data collected during a tensile test is discussed in Section 2.6.1. This use case's aim is to record the results of a test and produce a report.

A tensile test is where uniaxial load is applied to a sample of material in order to find the point at which the material deforms plastically and finally fractures. Raw tensile test data is normally in CSV format and is often converted to Microsoft Excel. To upload a tensile test data set, the user would create a folder for the data and copy the CSV file, possibly the Excel file, and can also choose to copy some scanning electron microscope images of the fracture surface. At this point, the user can enter some metadata about the test using the interface and use the reporting plug-in to produce a report. The report shown in Figure 4.13 shows tensile test data from a test performed on the material from Section 4.3.1. The fractography of the failure can be seen in the photographs.

To improve report content, users need to give the data set a description and enter metadata about the experiment in the interface. As with the material information use case, changing the data set type to a tensile test releases templates for that type and improves searching capabilities. The recommendation feature discussed in Section 4.2.3 uses the data set's type and relationships, so changing the data set type also enhances the recommendations made by the system as does defining a relationship between this data set and the material information data set.

#### 4.3.3 Long crack growth fatigue test

The data collected during a fatigue test is discussed in Section 2.6.2. This use case's aim is to record the results of a test and produce a report.

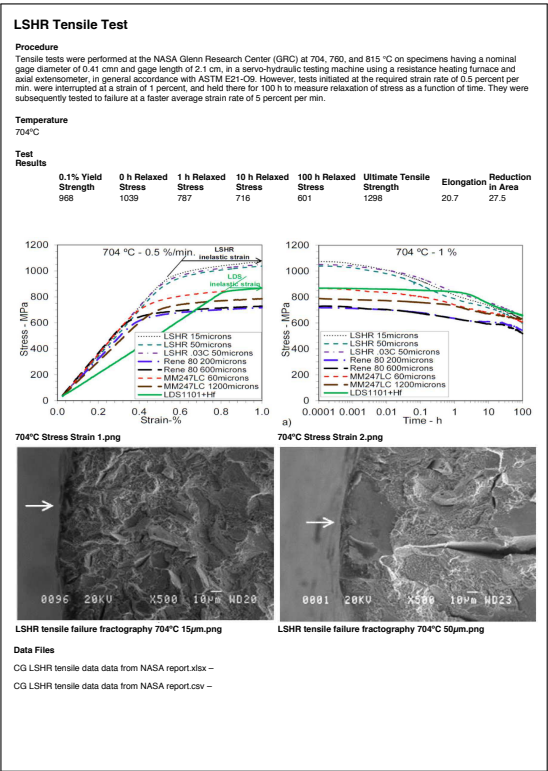


Figure 4.13: Tensile test report produced using the data set reporting plug-in.

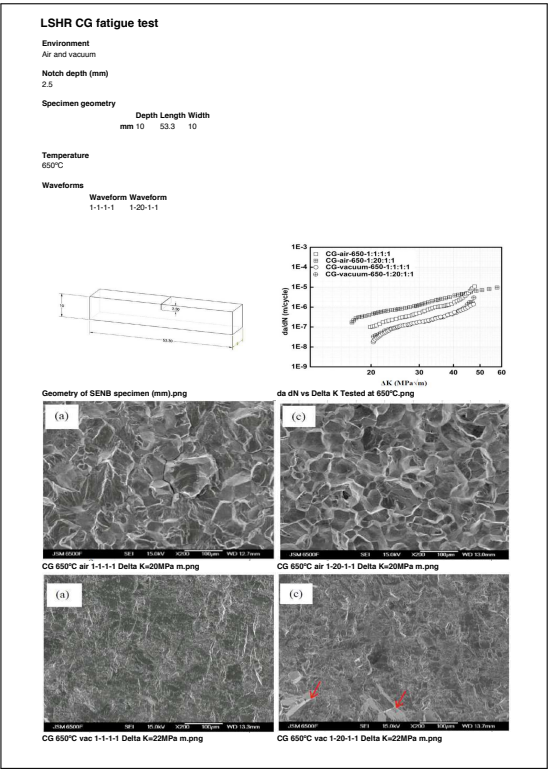


Figure 4.14: Fatigue test report produced using the data set reporting plug-in.



**Figure 4.15:** Three slices in the x-y, x-z and y-z planes extracted from a CT scan of a toy car using the CT image browser plug-in.

A fatigue test is designed to discover how resistant a material is to the growth of cracks. A sample of the material which already contains a crack is subjected to a cyclic load and the growth of the crack is monitored by passing an electrical current through the material and measuring how the sample's resistance increases as the crack grows. The raw data consists of a CSV file containing potential difference readings in relation to time. Its manipulation, such as smoothing the raw data and producing plots of  $da/dN$  (the rate at which the crack grows) against  $\Delta K$  (the range of stress intensity factor to be applied), is usually done with Microsoft Excel.

Data deposit of this test is similar to a tensile test: creation of a folder in the MDC file share and copying of raw CSV data and the processed Excel file. Metadata regarding the test can then be supplied. The report in Figure 4.14 is generated from data about a fatigue test on the material in Section 4.3.1.

#### 4.3.4 Microfocus computed tomography data

Engineers also commonly work with microfocus computed tomography data, perhaps having used a CT scanner to scan a failed material. To upload data to the MDC, the user again creates an empty folder in the file share, copies the CT data into it which could be several gigabytes, and can then use the MDC interface to populate any metadata.

The CT data browser plug-in can then allow users to view individual slices from the raw CT data file as seen in Figures 4.10 and Figure 4.15 so they do not need to download the entire file. Slice dimensions are guessed by parsing the name of the file.

The user should ensure they change the data set type to a CT scan, copy the accompanying .xtekct file containing metadata generated by the scanner, link to any data sets relevant such as a material data sheet and name their CT data files properly so slice dimensions can be detected.



Note that there is potential for data sets to be uploaded directly by the equipment to capture test image data automatically. We will show in the next chapter, in Section 5.2.6, how this can be improved further by capturing the metadata from the accompanying .xtekct file automatically.

## **4.4 Reliability tests**

### **4.4.1 Tests of the synchronisation service**

To test the synchronisation service we set up a number of tests as shown in Table 4.3. We were looking to prove that the service recorded the correct hash in its database and that the synchronisation service did not affect the data being stored. This was achieved using a set of bash scripts on a 16 GB Apple iMac (with model number MB953xx/A) running Mac OS X 10.6.8 connected to the MDC via Samba. The scripts created folders and files containing random data and checked the database records to verify the synchronisation service had operated correctly. Some statistics from the tests are shown in Table 4.4.

We created a file containing random data, hashing the data at the same time as it is generated to verify the MDC generates the correct metadata and does not affect the file in the file share. The command that was used to generate the random data is shown in Listing 4.2.

By piping the output through tee, the data could be written to the MDC at the same time as generating the SHA-1 hash to ensure data remained intact during the transfer.

We used `bsqldb` (from version 0.92.79 of the FreeTDS tools, installed from MacPorts) to query the SQL Server database to find the record in the database, sleeping while the synchronisation service generated the hash.

Finally, we checked the hash of the file in the MDC file share to ensure the contents of the file were written correctly and were not altered by the synchronisation service or during transfer to the server.

By running the tests in Table 4.3 we show that the service is reliable and can cope with differing sizes of data sets and files.

### **4.4.2 Testing the metadata database**

Parameters are nested by creating a record in the database linking two parameters. The depth of nesting is only limited by the number of records that can be created in a table but in practice will depend on the method of interfacing with the MDC. To discover limitations of the metadata component of the MDC, we created nested parameters up to 200 levels deep. Generation of the parameter editing web page remained sub-5 seconds for up to 100 levels with the simplest containers taking under a second; at 200 levels, generation of the parameter editing page took 19 seconds. The browsing experience in Safari 5.0.5, Firefox 3.6.18 or Internet Explorer 8 also suffered as they did not cope well with so many nested tables.

For our users, metadata depth is not expected to exceed five levels but optimisation steps will have to be taken if requirements change.

**Table 4.3:** Synchronisation service tests.

	Test description	Runs	Files	Errors
1	5 files: 1, 5, 10, 20 and 100 MB	200	1000	0
2	5 files of a random size $\leq 100$ MB	200	1000	0
3	Random number $\leq 100$ of 1 MB files	100	5365	0
4	100 files each 1 MB	100	10 000	0
5	1000 files each 1 MB	2	2000	0
6	5000 files each 1 MB	1	5000	0
7	Single 100 000 MB file	2	2	0
8	Single 400 000 MB file	1	1	0
<b>Other tests not scripted and not included in statistics (for testing of synchronisation service during development):</b>				
9	Single deposit of 152 data sets containing 3832 files (435 MB)			

**Table 4.4:** Synchronisation service test statistics.

Test data transferred	681.952 GB
Total number of files created	24368
Mean writing+hashing speed	3.7 MB/s
Mean reading+hashing speed	17.7 MB/s
Mean sync service hashing speed	4.9 MB/s
Largest file size	400 000 MB
Smallest file size	0 bytes <sup>1</sup>
Largest data set	5000 files

<sup>1</sup> For example, an empty ‘touched’ file in a Unix environment or a lock file in Microsoft Office.

**Listing 4.2:** Generating random data file, saving the file and printing its hash.

```
dd if=/dev/urandom bs=1m count=$TESTFILESIZE 2>/dev/null | tee \
↪ $TESTFILEPATH | openssl sha1
```

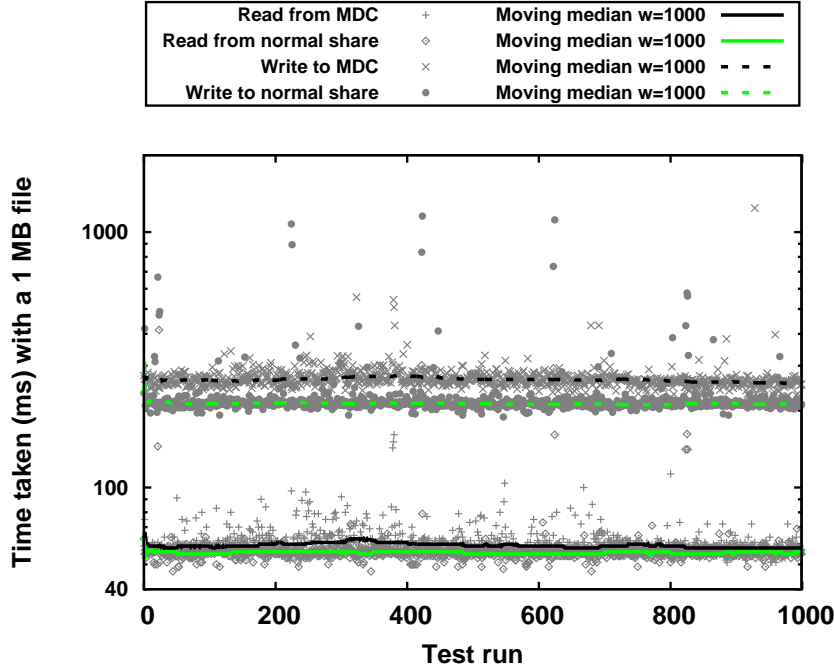


Figure 4.16: Performance results showing the effect of using the file monitoring service on a file share.

## 4.5 Performance tests

Figure 4.16 shows the effect of installing the file synchronisation service on a server using a simple moving median ( $w = 100$ ) of 1000 test runs. Without the file synchronisation service, the overall median time to write a 1 MB file was 213 ms and 56 ms to read it back (calculated over all 1000 test results). When the MDC service is enabled, the overall median time to write a 1 MB file was 263 ms and 59 ms to read it back, suggesting that there is an increase of 23% in the time taken to write data and a 5% increase in the time taken to read data when the file synchronisation service is monitoring a share. The overhead of hashing files and regularly checking the data does introduce a performance penalty.

This data was calculated using one run of test number 5 in Table 4.3 which generates one thousand 1 MB files containing random data using the test scripts, which pipe all data written and read through `openssl` to generate a SHA-1 hash. We have used the median for the overall results and a moving median for Figure 4.16 rather than the mean because there were some outlying data points which skewed the overall results (on both the MDC and normal share tests). For transparency, we also show all the data we collected, as well as the moving median. The prototype system and the testing system are not in the same buildings and are on shared networks; bottlenecks on the network, virtual machine host or even background tasks on the test client could be a factor.

## 4.6 Working with data in the system

Section 4.3 showed the flexibility of the model introduced in this paper through a number of use cases, and the combination of the file system and metadata database has been shown

to support a wide range of data, from basic information about a material to large, CT scan data. Our aim with the MDC was to ensure we could manage the wide range of data types discussed in Section 2.6 as well as relationships between data sets and various structured and unstructured metadata sets. In this section, we will discuss additional features for working with existing data that show there is further potential for such a system. We also highlight enhancements made to the system during user testing.

#### 4.6.1 Searching using the interface tools

Interface features have been added for users to perform simple searches against data sets in the system. For example, users can search for data sets by data set names, file names, and parameter names. Performance of this has been acceptable during user testing. Users can also retrieve lists of data sets by type (e.g. tensile tests) and collection from the interface.

#### 4.6.2 Data set suggestions

To increase awareness of other data sets, the recommendations feature in the interface suggests other data sets that might be of interest based on a data set's type and existing relationships. During the requirements capture stage, users requested this feature over standard search features and it has had good acceptance from our users in practice (Flanders et al. 2009; Soady and Scott 2011; Reed and Scott 2012), who are accustomed to features such as Amazon.com recommendations (Linden, Smith and York 2003).

#### 4.6.3 Data access API

The API provides the most functionality for working with metadata and this is how custom tools are normally expected to interact. Of course, it is possible to interact directly with the database, but the API provides several features useful for custom tools:

- Metadata related functions, for example:
  - Retrieving parameters, with or without nesting.
  - Retrieving an individual parameter's parents or children.
  - Adding or updating metadata, and creating hierarchical metadata (e.g. deleting a parameter tree).
  - Copying metadata between data sets.
  - Parameter validation, such as checking the Type and IsCompulsory fields.
  - System parameter related functions.
- Search functions, e.g. by ID, collection and finding data sets with specific parameters.
- Relationship management functions, e.g. data set links, collections and links between the data set type and templates.

The API will enable more powerful features to be built for users as they are needed. Some tools built upon the API have been demonstrated in Section 4.3 and additional tools will be demonstrated in the next chapter.

#### **4.6.4 SQL queries**

As mentioned previously, interacting directly with the database allows rudimentary queries to be created, such as finding lists of data sets by type, relationships and collections, lists of files within data sets, and even data sets with certain parameter names or values. It would be feasible to perform queries with SQL that take into consideration the nesting of metadata, although it would be more optimal to use the API or an interface tool that has this capability.

#### **4.6.5 API example queries**

To demonstrate the API, and illustrate some of the potential of storing data in such a way, we will now provide a few examples of its use. The API, described in Sections 4.2.3 and 4.6.3, provides some wrapper functions to perform common tasks and it also provides a database model (using the Entity Framework features in Microsoft .NET) to allow custom queries to be constructed. We will show examples of both types of query, and use C# and Python to show interoperability possibilities.

The examples use Microsoft .NET Framework version 3.5, and the Python examples were built with IronPython 2.7.4. To keep them concise, any preamble code for adding namespaces and references to external assemblies, and any sensible error checking that would be necessary in production has been excluded. The API uses the term ‘experiment’ to signify a data set which was highlighted in Section 4.2.2, reflecting the project’s provenance in storing test data for materials engineers.

#### **Copying parameters between data sets**

Listing 4.3 shows a simple Python script example of how to retrieve data sets by ID from the database and then use the API to copy the metadata from one data set to another. This duplicates the parameter definitions and their structure. Parameters are also used by the system to record information such as a data set’s thumbnail and these can optionally be excluded. Internally, the `CopyParametersToExperiment` function records the IDs of each source parameter, as discussed in Section 4.2.2.

#### **Data set collections and simple nested parameter operations**

Listing 4.4 shows a slightly more complex Python example of recursing through nested parameters and printing them to the screen in a tree-like format. The `GetExperimentsByCollectionID` function is used to retrieve several data sets, and then the `GetExperimentTopLevelParameters` function retrieves the top-level parameters of each data set. The Python `print_parameter_tree` function then uses recursion and the `GetChildParameters` API function to print any nested metadata.

**Listing 4.3:** Copy parameters between data sets (Python).

```

1 with MaterialsMetadataContext(connection_string) as dbcontext:
2     material_n18 = dbcontext.GetExperimentByID(16)
3     material_new = dbcontext.GetExperimentByID(291)
4     material_new.CopyParametersToExperiment(material_n18, copySystem=False,
↪     dbcontext=dbcontext)

```

**Listing 4.4:** Print parameters in a tree for all data sets in collection 1 using a recursive function (Python).

```

1 with MaterialsMetadataContext(connection_string) as dbcontext:
2     # Define a function to recurse through nested parameters
3     def print_parameter_tree(parameters, level=0):
4         prefix = "\t"
5         for i in range(0, level):
6             prefix = prefix + "\t"
7         for p in parameters:
8             print prefix + p.Name, ":", p.Value
9             child_parameters = p.GetChildParameters()
10            if child_parameters.Count > 0:
11                print_parameter_tree(child_parameters, level + 1)
12
13    # Retrieve data sets from collection 1
14    datasets = dbcontext.GetExperimentsByCollectionID(collectionID=1,
↪    includeDeleted=False)
15    # Print info from all data sets
16    for ds in datasets:
17        print "Data set: ", ds.Name
18        parameters = dbcontext.GetExperimentTopLevelParameters(ds.ID,
↪    includeSystem=False, parameterName=None)
19        print_parameter_tree(parameters)

```

**Listing 4.5:** Obtain a list of data sets with a chromium value of greater than 10 (C#).

```

1 using (MaterialsMetadataContext dbContext = new MaterialsMetadataContext(
  ↳ connection_string))
2 { foreach (Dataset ds in dbContext.GetExperimentsWithMatchingParameter("
  ↳ Composition"))
3   { // Retrieve Composition parameter (taking first one if more than one
  ↳ if found)
4     var compositionParameter = ds.LookupParameters("Composition").
  ↳ FirstOrDefault();
5     if (compositionParameter == null)
6       continue;
7
8     // Look up child chromium parameter (taking first one if more than
  ↳ one is found)
9     var chromiumParameter = compositionParameter.GetChildParameters("Cr")
  ↳ .FirstOrDefault();
10
11     // Only use it if the parameter matches its definition
12     // (IsValid checks Value against Type and IsCompulsory fields)
13     if (chromiumParameter == null || !chromiumParameter.IsValid)
14       continue;
15
16     float chromiumValue;
17     if (float.TryParse(chromiumParameter.Value, out chromiumValue) &&
  ↳ chromiumValue > 10)
18       Console.WriteLine("Material '{0}' has {1} {2} chromium.", ds.Name,
  ↳ chromiumParameter.Value, chromiumParameter.Unit);
19   }
20 }

```

### Data set filtering and parameter querying

Listing 4.5 uses C# to find data sets that contain the Composition parameter with the GetExperimentsWithMatchingParameter function and then prints the value of the nested Cr parameter only if it has a value greater than 10. Note the use of the IsValid property which checks IsCompulsory and Type fields and ensures Value contains appropriate data.

### Data set information lookup and metadata editing

Listing 4.6 is an example written in Python and inspired by the eCrystals project in the Chemistry discipline. It shows how to retrieve data sets with a specific property and perform some action on them.

Firstly, it retrieves all data sets that contain a parameter with a name of Crystal system and a value of triclinic. It then iterates through each data set, but only if the data set has not been seen before (by checking for the Test script executed date parameter). Information about the data set is then printed to the screen and then the Test script executed date parameter is created to exclude it from future runs of the script.

The action performed in this example is to retrieve the file path and web page of the matching data sets and simply print to the screen, but clearly there is potential for files to be created, read or modified, utilising the information that has been retrieved from the database,

**Listing 4.6:** Find data sets with a specific property and perform some action on them (Python).

```

1 with MaterialsMetadataContext(connection_string) as dbcontext:
2     # Find data sets with a specific property
3     datasets = dbcontext.GetExperimentsWithMatchingParameter("Crystal
↪ system", "triclinic")
4     # Convert matching data sets to an array and loop around them
5     for ds in Enumerable.ToArray[Experiment](datasets):
6         # Check if we've visited this data set before
7         visitedparameter = ds.LookupParameter("Test script executed date")
8         if visitedparameter is None:
9             # Perform some action on the data set
10            print ds.ID, ds.Name, "-", ds.GetUNCPath(dbcontext), ds.
↪ GetDetailsWebPage(dbcontext)
11            # Record that we have visited this data set to skip it next time
12            ds.SaveParameterValue(dbcontext, "Test script executed date", time.
↪ strftime("%c"), create=True)
13            dbcontext.SaveChanges()

```

**Listing 4.7:** Find recommended data sets related to data set 16, list some information and create a link to them (Python).

```

1 with MaterialsMetadataContext(connection_string) as dbcontext:
2     tensile_test_dataset = dbcontext.GetExperimentByID(16)
3     related_datasets = dbcontext.GetRelatedExperiments(tensile_test_dataset
↪ .ID, count=5, findDeleted=False)
4     print "Related data sets:"
5     for ds in related_datasets:
6         # Print data set information, the data set's UNC path and the data
↪ set's web page
7         print ds.ID, ds.Name, "-", ds.GetUNCPath(dbcontext), ds.
↪ GetDetailsWebPage(dbcontext)
8         # Link data set 16 to recommended data set
9         dbcontext.CreateExperimentLink(tensile_test_dataset.ID, ds.ID,
↪ saveChanges=True)

```

or perhaps to generate a script for execution on a high-performance computing cluster that stores its results in the appropriate data sets.

### Data set recommendations and relationships

As discussed in Section 4.2.3, the system can suggest related data sets based on a data set's existing relationships. This Python example, shown in Listing 4.7, shows two main features: retrieving a data set's recommended data sets and how to create a new relationship between two data sets. Firstly, it retrieves the information for a specific data set (16) from the database and then it requests five recommendations by calling the `GetRelatedExperiments` function. It then prints each of these to the screen and then calls the `CreateExperimentLink` function to create a link between data set 16 and each new recommendation.

While this is a synthetic example, it succinctly demonstrates the creation of links between data sets and the data set recommendation feature.



**Listing 4.8:** Using LINQ instead of the API to search for data in the database (C#).

```

1 using (MaterialsMetadataContext dbcontext = new MaterialsMetadataContext(
  ↳ connection_string))
2 { int ctscanType = 2;
3   var datafiles =
4     // Retrieve data sets filtered by type
5     dbcontext.Experiments.Where(ds => ds.ExperimentType.ID == ctscanType)
6     // Retrieve all files in the matching data sets
7     .SelectMany(ex => ex.ExperimentDataFiles)
8     // Filter the files by name
9     .Where(df => df.Filename.Contains(".xtekct"));
10
11   foreach (ExperimentDataFile df in datafiles)
12   { // Load data set information from database
13     df.ExperimentReference.Load();
14     // Parse file (for brevity, we assume file exists and is valid).
15     FileParser_XTEKCT filedata = new FileParser_XTEKCT(df.GetFullPath());
16     foreach (KeyValuePair<string, string> pair in filedata.Properties)
17     { //Create parameter
18       ExperimentParameter newParam = new ExperimentParameter() { Name =
  ↳ pair.Key, Value = pair.Value };
19       dbcontext.AddExperimentParameterToExperiment(df.Experiment.ID,
  ↳ newParam);
20     }
21     dbcontext.SaveChanges();
22   }
23 }

```

### Custom queries using the Entity Framework database model

The final example, in Listing 4.8, is written in C# and shows a more complicated query using a LINQ construction. LINQ (Language Integrated Query) extends the syntax of C# and Visual Basic to allow powerful querying and updating of data (Microsoft 2014c). This type of query is currently not possible with IronPython and the API, but something similar would be possible with Visual Basic .NET. The LINQ statement is translated into an appropriate SELECT SQL statement and executed against the database, and the Entity Framework then translates the query into a list of objects which can then be used in the rest of the code.

This example retrieves a list of all of the data files in the system that have .xtekct in their name from data sets that are CT scans. It then uses a file parser to parse each file, extract the properties, and create appropriate name-value metadata in its data set. We will show a plug-in which uses an .xtekct file parser in the next chapter, in Section 5.2.6.

This program is operating at two levels: with the files on the file system and with the metadata in the database. It is using the metadata to drive which files will be read, then those files are parsed and the results are stored back in the database.

## 4.7 Discussion

The model of file store, metadata database and example interface hosted in Microsoft SharePoint has been built as an output of this project and tested with a number of use cases: a

material information sheet, a tensile test, a fatigue test and 3D densitometry data. This section evaluates the system looking at the metadata database, data store and interface components.

#### 4.7.1 Feedback from user testing

The MDC system was piloted by researchers in the materials engineering department. Most of the feedback involved requests for new reporting features or improvements to the user interface. The following list contains the weaknesses found by users; some of these will be addressed in the next chapter and for the others we will discuss possible solutions:

**Reordering of metadata entries, especially on reports:** The ability to reorder metadata entries will involve a modification to the database schema to record the parameters' order which will be shown in the next chapter.

**More advanced reports to include data across data sets and ordering of items in reports:** The reporting plug-in could be adapted to allow this feature. We will show more advanced reporting in Chapter 5.

**Browsing data sets with large numbers of files or parameters can be slow:** Improvements such as hiding of files, browsing of files by subdirectory (instead of listing all files in a data set), and optimisation of the prototype code in some components (such as not listing all parameters, files, plug-ins and all other data set information on the same page and reducing the number of round trips to the database) would improve performance.

**All data sets must be deposited together with other users':** This requirement was not welcomed by users who quickly found it difficult to create new data sets, such as only one 'Tensile test' being permitted. The improvements made in the next chapter will remove this limitation.

**Upload of data using the web site:** While the main attraction of our approach is the use of the file system for data set upload, we found that some users still desired the ability to upload data via the system's interface. This will be discussed in Chapter 5.

**Ability to create more complex tables in reports:** One of the reports includes a simple two row table consisting of one row of 'Name' entries and one row of 'Value' entries. Some users wanted more complex tables with multiple rows and multiple columns. A new interface component could allow users to encode more complex tables using parameters to represent the cells in the table – no changes to the database schema would be required. An example of how this might be encoded is shown in Tables 4.5(a) and 4.5(b).

The ability to incorporate features such as these and adapt the system in response to feedback demonstrates the flexibility of the architecture.

**Table 4.5:** Encoding larger tables using the MDC's metadata database.

(a) Parameters to encode a more complex table.			(b) The resultant table.		
Parameter	Name	Value	Element	Wt %	Description
1	Column headings	—	Co	15.4	Cobalt
1.1	Element	—	Cr	11.1	Chromium
1.2	Wt %	—			
1.3	Description	—			
2	Row	—			
2.1	Element	Co			
2.2	Wt %	15.4			
2.3	Description	Cobalt			
3	Row	—			
3.1	Element	Cr			
3.2	Wt %	11.1			
3.3	Description	Chromium			

#### 4.7.2 Metadata and data storage

Storing the data set metadata in the way we have has permitted the wide ranging data types required in the MDC to be supported with a very simple schema. We have also successfully employed the metadata storage approach to store configuration data for the MDC system and used templates to ensure the configuration data sets are created correctly.

Due to this very simple database structure, filtering and searching even with large data sets should be scalable as complex queries with many joins are not involved. In some cases the EAV approach may introduce a performance penalty but for a user-driven data repository, where searches are generally simple, any delays should not become a disadvantage. For intense data analysis, a database structure tailored to the application is likely to give better performance.

Modifications to the database may be necessary in order to improve the toolset of the MDC, such as the ordering of metadata parameters requested by users of the prototype.

The approach of storing the data in a file system provides the ability to seamlessly take advantages of developments in storage technology as they become available in the future, such as cloud storage and improved local file systems.

There is an impact on the performance of using a share monitored by the MDC file synchronisation service (5% on reading, 23% on writing), caused by the additional overhead of collecting metadata including hashes of incoming data as discussed in Section 4.5.

Some of the disadvantages of using an EAV inspired approach were discussed in Section 2.7.3. This included an inability to use mandatory values and data types and the difficulty with ensuring attribute names are used consistently. These issues are partially mitigated with a number of features that have been discussed in Section 4.2.2. The templating features allows predefined metadata sets to be copied onto data sets ensuring metadata is completed consistently, and by recording where parameters were copied from it is possible to repair or highlight names that had been mistakenly changed. The additional fields we have included

in the parameter table allow us to support mandatory values and data types at the interface level. However, it is important to note that the user is in control of their metadata as this is a system for user driven data management and in early prototypes of the system, user testing showed that enforcing mandatory values or types frustrated users (Reed and Scott 2011; Reed and Soady 2010; Hawkins and Scott 2012). Therefore the interface ensures users are aware of any discrepancies but does not enforce anything, ensuring they can always produce the reports that they require or record the metadata needed for their situation, even at the cost of consistency of user reports.

The effect of this is still unknown. User feedback has so far shown that they are content with the metadata features provided, other than suggested improvements already discussed in Section 4.7.1. It is postulated that parameter names will remain consistent for users who use the templates, but the system is flexible enough to allow the user to deviate from the template if required.

### 4.7.3 Materials Data Centre interface

The interface components allow users to manage the metadata surrounding their data sets, produce reports, execute plug-ins and create relationships between data sets for management and to obtain data set recommendations. As a consequence of using Microsoft SharePoint as the underlying interface technology, users are also able to produce their own landing pages for their data by using the provided MDC web parts, by linking to an automatic report or, for advanced users, creating a custom web part that can present the data in a meaningful way. Users could also manage their own project sites, including the ability to use blogs and wikis.

## 4.8 Summary

In this chapter, we presented a model consisting of a metadata database combined with a file system, developed for the management of materials engineering data, and have shown its use with a number of use cases and an example interface which was developed to be used with Microsoft SharePoint to take advantage of some of its features such as user-created wikis. The combination of a flexible database for metadata and file system for storing images, raw data and documents has been able to cope with many of the different types of materials data and disconnecting the file storage and metadata from the technology chosen for the interface introduces much more flexibility to the system.

The linking and typing of data sets helps the system to recommend related work to the user which has a lot of potential to improve the dissemination of data sets.

Allowing users to upload data sets via a file share simplifies data set upload and has huge potential for capturing data sets in materials engineering and at the research group level, making it easier to transfer to a data set archive at a later date because the data and some accompanying metadata was at least captured. The approach taken is suitable for storage of any data files and metadata that can be recorded against a set of files. We have shown this flexibil-

ity with a number of use cases, suggesting that the architecture of the MDC may be flexible enough to be used in disciplines other than materials engineering.

We have shown several examples of using the API, mainly with Python but also a couple of C# examples. The choice of mainly Python examples was intentional as Python is an appropriate scripting language, allowing light-weight code to be produced quickly from a command prompt, particularly important when dealing with many data sets and rapid analysis or reporting is required. The more powerful, interface components would be written in C# where the entire functionality of the API and Entity Framework database model can be utilised. These examples have shown the power and flexibility provided by the API. Listing 4.6, which showed querying data sets holding crystal data from Chemistry based on their Crystal system property, also shows that there is potential for generalising the framework for use in other disciplines.

In order to support heterogeneous data, this model is lacking in several areas. The requirement for all data sets to share one folder – a remnant of choosing SharePoint’s document libraries to store data sets in the EP2DC project – severely limits the data centre because data sets can only be created with a name that has not been used before (for a user-driven data centre, this is not very user friendly). Secondly, users have no control over the security of their data sets. Thirdly, data repositories tend to use standard protocols for interaction; although we designed a REST web service for the early EP2DC project, we should demonstrate our model is flexible enough to support other standards. Finally, feedback on the prototype system from our materials users has generally been on additional tools or new reports and some ideas on improving the usability of the interface. We will look at these problems in the next chapter.

The generic nature of the design led to a new name for the project: the Heterogeneous Data Centre (HDC). The next chapter will look at improvements that the HDC project made to the MDC and will show its use with medical data while continuing to support materials research. We will also investigate using an RDF representation of the data to increase the power of queries that can be performed and to show the HDC can be adapted to make it support accepted standards.

## Chapter 5

# A model for managing heterogeneous data

---

The work described in this chapter has been published as:

Mark Scott, Richard P. Boardman, Philippa A. Reed and Simon J. Cox. 2014. ‘Managing heterogeneous datasets’. *Information Systems* 44 (August): 34–53.  
doi:10.1016/j.is.2014.03.004

---

## 5.1 Introduction



UPON COMPLETION of the Materials Data Centre project, a number of limitations were identified which were discussed in Section 4.8. This chapter discusses how these were addressed to investigate using the model for the management of heterogeneous data. With this in mind, we will take a couple of use cases from medical science, specifically human genetics. Additional features will concentrate on four areas:

1. Straightforward data deposition process.
2. Metadata and metadata tools.
3. Data set discoverability.
4. Data set tools adding value to users, to encourage uptake and induce users to provide data and metadata.

We will assess the system against these principles in Section 5.4.

## 5.2 Improvements since MDC

The main weaknesses identified in Chapter 4 were the requirement for all users to share the same folder for data sets, the lack of control users have over the security of data sets (administrators of the system were required to grant access to data sets), a number of possible interface improvements and the need to show that the model can be adapted to support accepted standards. In this section we will show the following additional features, which address these deficiencies:

- Per-user and per-group data sets
- User-driven data set security
- Ordering of metadata via the data set's details page (reports and other tools can observe the parameter's position if applicable)
- Plug-ins:
  - Metadata import
  - Automatic plug-in execution
  - Delimited file browser
  - Stats file browser
  - Data file graphing
- Interface improvements:
  - Adding data sets and files from the interface
  - Improved reporting and tools
- Support for RDF-compatible tools, demonstrated with SPARQL

We will use the improved data model to show the support of data from the medicine discipline with examples from human genetics in Section 5.3.

### **5.2.1 Final metadata database schema**

The complete set of database tables and fields for our data management model is shown in Figure 5.1. We discussed the metadata database in Section 4.2.2. Modifications made since this – the security additions discussed in Section 5.2.3 and the metadata reordering discussed in Section 5.2.5 – have been shown in a different colour.

### **5.2.2 Per-user and per-group data sets**

Folders can now be placed in per-user or per-group structures rather than one folder shared across all users. This was accomplished by modifications to the file system monitor component of the HDC with the introduction of a configuration option (`BasePathDepthIn-WatchFolder`) to specify how many levels of nesting are permitted before a folder creation is recognised as a data set. This still allows the system to be configured in the historical approach of one folder for all data sets with a value of 0, a per-user approach with a value of 1 and a per-group approach with a value of 2.

Modifications to the database schema and interface components were not necessary, which are only concerned on the existence of the data set not how it was found by the file system monitor. This implies data set recognition and ingestion is possible with other approaches including: the creation of a file or folder with a specific name inside a data set to activate the

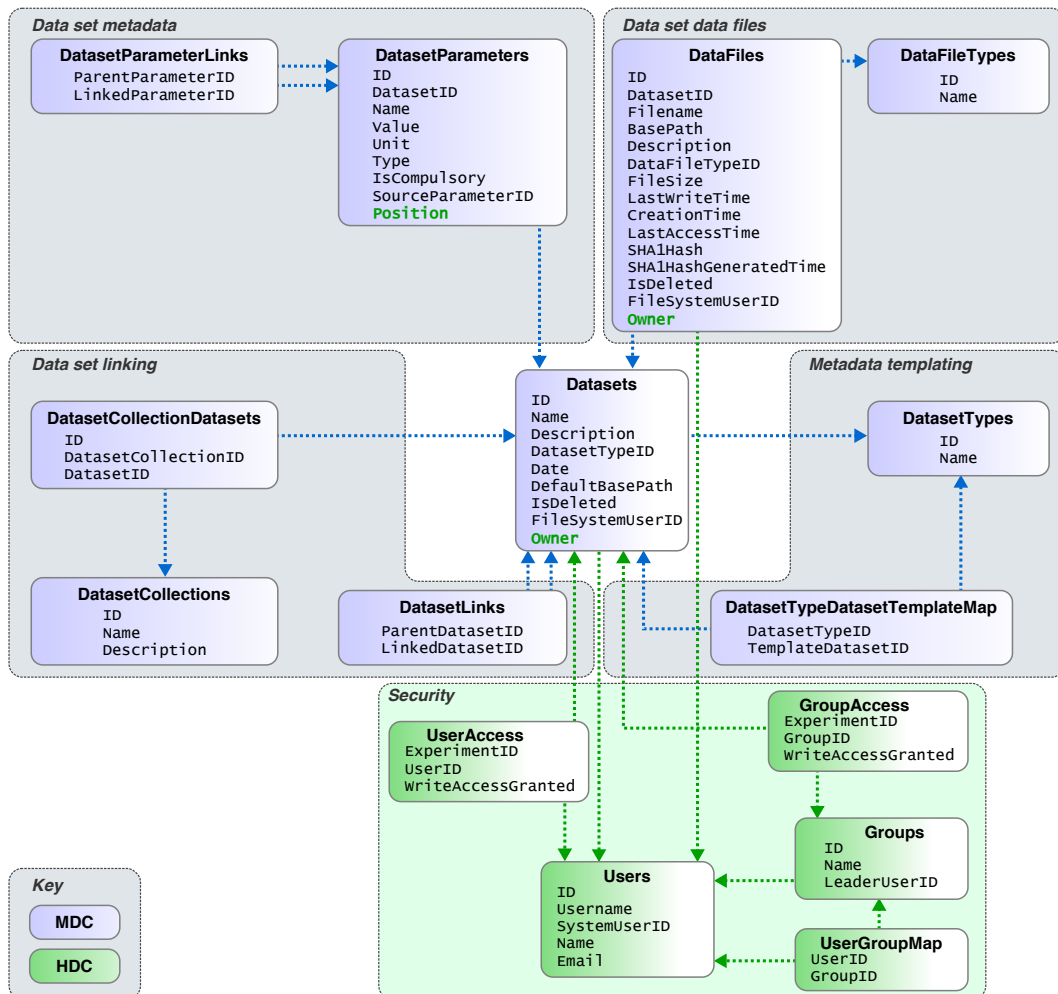


Figure 5.1: Metadata database schema in data management model.





Figure 5.2: Data set details page, (1) security section.

ingestion manually at whichever level it is stored; the use of the web interface to manually select which folders on the file system to ingest as data sets; or the creation of a client-side tool to select which folders are to be treated as data sets. These approaches avoid having to enforce a data centre-wide setting, ensuring the system is entirely user driven and allowing the file system component of the HDC to perform much more like a tradition file server.

### 5.2.3 User driven data set security

Easily controlling access to a data set can be key to the dissemination of data but many users do not know how to edit file system access control lists or have not been granted the permission to do this by their systems administrators.

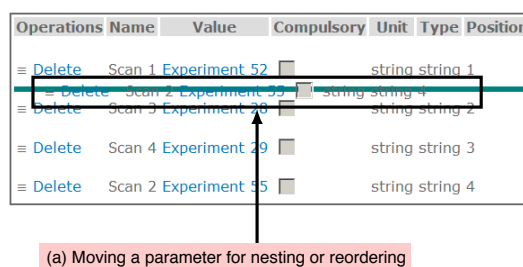
The database tracks which users have access to which data sets, with data sets only accessible to their owners and system administrators to begin with. Extra access can be added or revoked with the system propagating these permissions down to the file system on their behalf. When a change is made to the database, this is passed back down to the file system by the file system monitoring service. This involved the addition of an interface component (see Figure 5.2(1)), the modification of the database to store the security settings (see the ‘Security’ section in Figure 5.1), and improvements to the file system monitor to perform the security modifications based upon the database.

### 5.2.4 Data set details page redirection

When checking a data set, the file system monitor now creates an .html file in the tracking file’s folder which can be opened in a web browser to redirect to a web page for editing the data set’s metadata. This was to make it quicker and easier for users to edit the metadata of their data sets.

### 5.2.5 Metadata reordering

To support metadata reordering, as requested by users and discussed in Section 4.7.1, the DatasetParameters database table was altered to include the Position field, and the metadata interface and reports were improved to support this feature; Figure 5.3 shows metadata reordering in the interface by dragging a parameter and dropping it in the position required.



Operations	Name	Value	Compulsory	Unit	Type	Position
≡ Delete	Scan 1	Experiment 52	<input type="checkbox"/>		string	string 1
≡ Delete	Scan 2	Experiment 55	<input type="checkbox"/>		string	string 4
≡ Delete	Scan 3	Experiment 38	<input type="checkbox"/>		string	string 2
≡ Delete	Scan 4	Experiment 29	<input type="checkbox"/>		string	string 3
≡ Delete	Scan 2	Experiment 55	<input type="checkbox"/>		string	string 4

(a) Moving a parameter for nesting or reordering

**Figure 5.3:** Data set metadata editing (reordering and nesting). In this figure, the line between the top two rows of the table shows that the dragged parameter ('Experiment 55', highlighted by the black box) will be inserted between the top two rows. Nesting is shown in a similar way: by colouring the row that will become the new parent of the parameter being dragged.

## 5.2.6 Plug-ins

### Metadata import

.xtekct files, generated by the CT scanner, were copied as part of the data set in the CT data use case in Section 4.3.4 and were parsed using example code in the Entity Framework API example in Section 4.6.5. This section describes the parsing plug-in.

The .xtekct metadata import plug-in parses files produced by a CT scanner in the .xtekct format and creates metadata entries in the data set containing whatever information was saved by the scanner about the conditions of the scan. We will discuss the significance of this in Section 5.4.1.

This plug-in can be used to import any data files that follow the key=value format. Nesting up to one level is supported by inserting a line containing the parent parameter's name in square brackets before a group of child parameters. This format is simple enough for some users to create their own files in this format to import large amounts of metadata rather than using the web interface to enter metadata manually.

### Automatic plug-in execution

Section 4.2.5 introduced data set and data file plug-ins, which were activated by the user from the interface. A third type of plug-in was added to the file system monitor, automatically executing on deposit of a file in a data set based upon a file's extension. This has advantages for automatic metadata ingestion. For example, the .xtekct metadata plug-in introduced above has been enabled to activate automatically when a file is uploaded with the .xtekct extension is uploaded.

### Delimited file browser

The delimited file browser plug-in permits viewing of a delimited text file and allows users to select which columns and rows they want to view – see Figure 5.4. Additionally, this subset of data can then be: exported to a new data file; a link to reproduce the report can be saved as a parameter of the experiment; or the selected data can be imported as metadata.

File has 22778 lines. Showing 10 lines starting at line 5.

Start row:

Max rows:

Column delimiter ('\\t'=tab):

☒ Use header row on line:

[Refresh](#)

**Field selection:**

☐ SAMPLE ID

☒ CHR

☒ LBP

☒ RBP

☐ Genomic sequence content +/- 50bp from LBP and RBP, repeat masked DNA in lower case

☐ Flag

	CHR	LBP	RBP
5	chr5	112204224	112204224
6	chr5	112204458	112204458
7	chr5	112205070	112205070
8	chr6	38758606	38758606
9	chr6	139605607	139605607
10	chr12	109837939	109837939
11	chr6	32114296	32114296
12	chr3	75868915	75868915
13	chr19	40528370	40528370
14	chr19	53951341	53951341

[Save link to report as parameter](#)

[Export report to file](#)

[Import selected data as metadata](#)

[Generate X Y Graph](#)

**Figure 5.4:** Browsing the contents of a tab delimited file. The user can choose which columns and rows to view, and can save the report or export the data as a new data file (choosing whether to save in the HDC file share is left up to them).

The delimiter can be customised so the plug-in can be repurposed for, e.g. tensile test data (comma-separated) as discussed in Section 4.3.2; we will show the plug-in in use with medical data (tab-separated) in Section 5.3.1.

### Data file graphing

The fields in a delimited file can be selected as X and Y values for creating a scatter plot<sup>1</sup>. Multiple files can be loaded by the tool, permitting quick creation of plots comparing values from separate files in separate data sets. The importance of this is demonstrated by giving an example with tensile test data (introduced in Section 4.3.2): the scatter plot plug-in can be employed for viewing the raw CSV data in tensile test data files and comparing to other tensile tests as shown in Figure 5.5. In this case, we can see how one sample failed earlier than another.

### Comparing data set metadata by graphing

Figures 5.6 and 5.7 show how the user can compare the metadata stored across multiple data sets by using the graphing plug-in. For example, the user can compare the composition of materials. Once the user has selected a data set, the plug-in only allows selection of other data sets with common parameters and secondly, will only permit metadata items to be selected that are common to the selected data sets. The graph can be saved as a link for reproduction later.

1. Charts are produced using the HighCharts charting library (<http://www.highcharts.com/>)

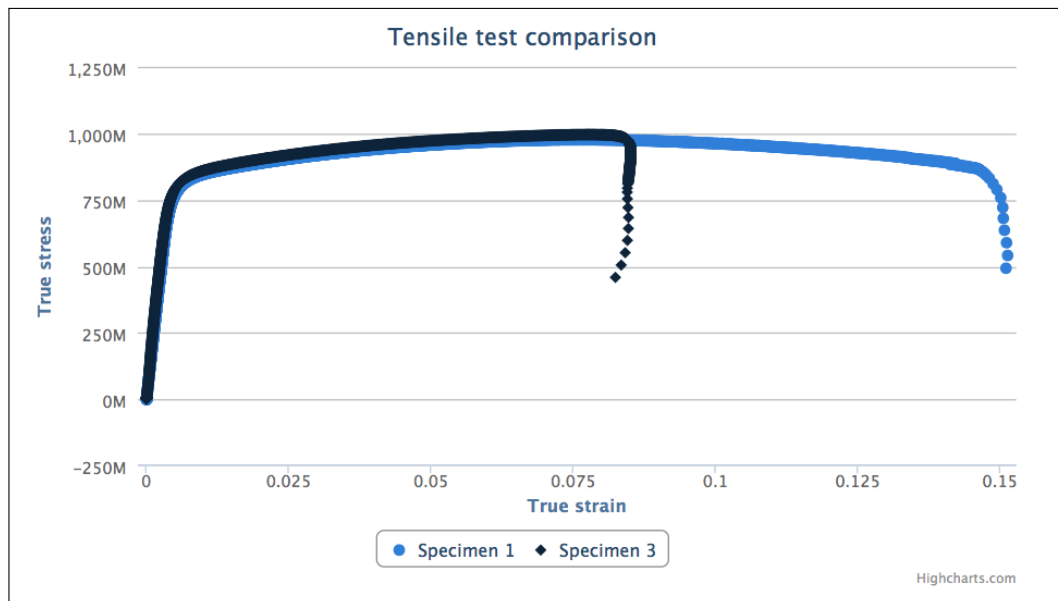


Figure 5.5: Scatter plot comparing two tensile tests from two delimited data files.

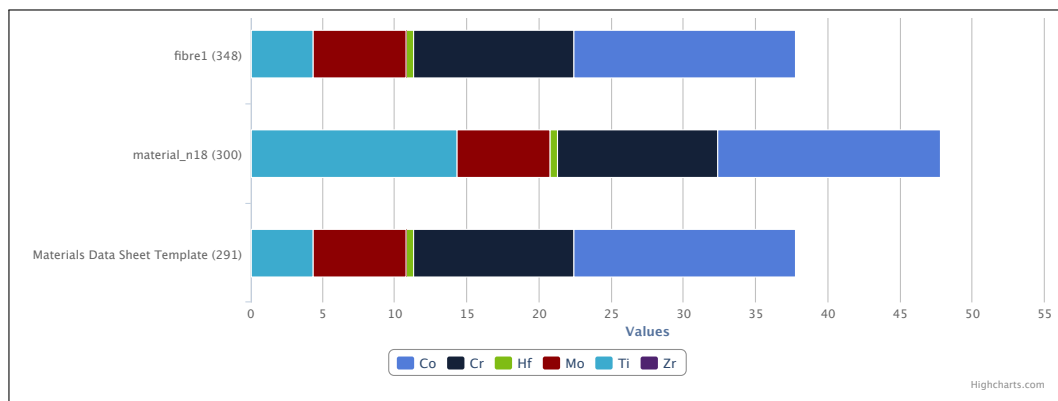


Figure 5.6: Comparing the compositions of materials using the graphing feature (bar chart).

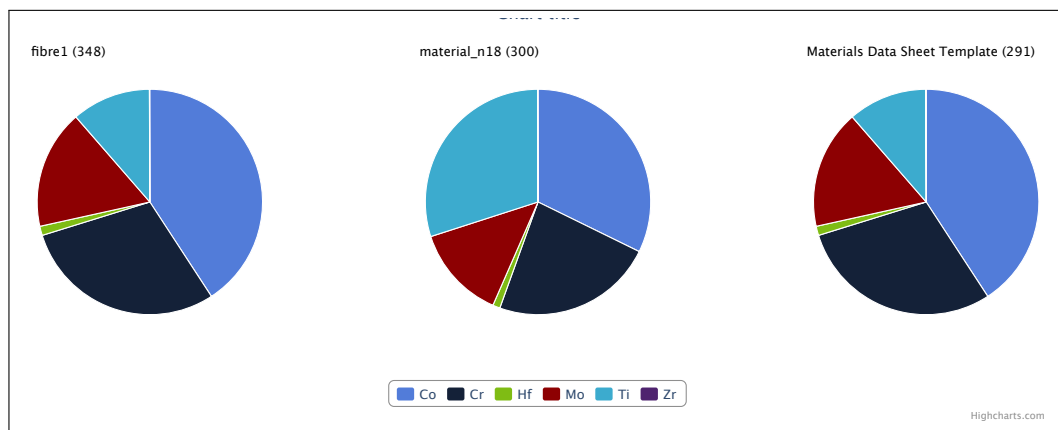


Figure 5.7: Comparing the compositions of materials using the graphing feature (pie chart).

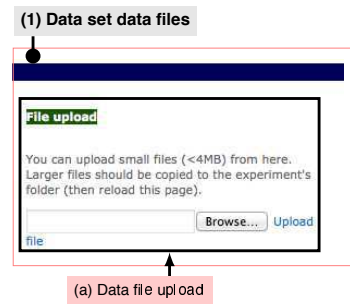


Figure 5.8: Data set details page, showing (a) data file upload via web site.

## 5.2.7 Interface improvements

### Improved reporting across data sets

Reporting was improved to allow selection of parameters and files across data sets. This formats a data set's metadata entries and image files in a suitable format. If required, data and metadata from other data sets can also be included making it possible for users to create ad-hoc reports from their data sets instead of relying on the automatic reports for individual data sets demonstrated in Section 4.3.

### Adding data sets and files from the interface

The preferred method of uploading data sets into the system is through the file system. This supports files of any size (limited by the file system), the user can upload many files at once and can control where the data set is stored. However, during testing, users requested the ability to do more within the interface (Reed and Scott 2011) so the ability to create data sets and upload files from the interface was added (see Figure 5.8(a)).

The uploading of files via the interface is limited by the web server used – in our case, Microsoft SharePoint and Internet Information Services defaults to 50 MB (Microsoft 2009a, 2013c, 2013a). This can be customised of course, but after discussion with users, we felt this was a sensible starting point; the uploading of large data files via a web page increases the load on the web server and without some sort of file upload manager that can resume failed uploads and can ensure large files are sent reliably, the user experience may suffer whilst waiting for the browser to complete the operation.

## 5.2.8 RDF representation of data and metadata

By producing an RDF/XML representation of the data sets, their metadata and their relationships, tools that can understand this format can be used against the database; for example, SPARQL (Prud'hommeaux and Seaborne 2008), a language for querying RDF graphs, can be employed to perform advanced searches across the data.

Two RDF representations of the HDC data are produced: one with all parameters on a data set 'flattened' and one with the parameter nesting preserved. The flattened version is probably enough for most purposes as the SPARQL queries required will be much simpler as they will not need to take into account the nesting.

This has the potential to allow user-level queries based on data set type, relationships, metadata values and even metadata nesting, without interacting with the database directly. The RDF formats are well recognised standards and good at representing relationships with ‘RDF triples’ (which is able to represent the EAV metadata from our model adequately), the provision of an RDF representation of the data facilitates interaction with other existing tools. For example, a list of data sets that are of a specific type can be retrieved by filtering the data sets by `dcterms:type`. Note that `dcterms:type` is a property from the Dublin Core metadata set. While we do not enforce any metadata requirements on the user, some properties can be mapped across to the Dublin Core properties, such as the data set’s ID, name, description, type, date and relationships.

The potential for this in a local repository is demonstrated in Section 5.3.2 where it is shown that users can use metadata and the relationships between their data sets to find data sets with specific properties.

## 5.3 Use cases

### 5.3.1 Human genetics use case

The wide range of data produced by materials engineers meant the Heterogeneous Data Centre had potential for managing data produced by other disciplines. We looked at medical data to test its flexibility.

Researchers into human genetics take a DNA sample and analyse it using a sequencing machine which produces short sequence *reads* representing the sequence as millions of fragments. The fragments represent the sequence of 3 billion nucleotides producing a data set of up to 50 GiB. The most cost-effective strategy at this time is to only sequence the protein coding regions (exome) which is about 1% of this data. The data generated is in a text based format known as FASTQ (Cock et al. 2010).

The FASTQ data is pre-processed by a software package called Novoalign (Novocraft 2013; Ruffalo, LaFramboise and Koyutürk 2011) to align the short reads against a complete human genome reference sequence. The aligned data can then be processed using the software tools in the Genome Analysis Toolkit (McKenna et al. 2010) to identify regions that are different from the reference data set, or *variants*.

Comparing sequences from two subjects’ samples will identify thousands of differences; the majority of which make no difference to the protein that the gene codes for. *Synonymous* differences do not change the protein, but *non-synonymous* differences will. A DNA sequence with a *frame shift* mutation codes for a protein that is considered to be non-functional. However, an average adult will have a large number of these differences and still be healthy and research has only just begun to understand the variants and how they can cause diseases.

Filtering against genome databases permits known diseases to be identified by comparing the variants identified with variants that are known causes of disease, and the effect a variant may have on the protein can be calculated by tools such as *SIFT* (Sorting Intolerant From Tolerant).

Microsoft Excel is frequently used to manipulate the variance data produced which is in tabular form with one row per variant, including data such as the location of the variant in the protein, the reference nucleotide's value and the variant's, the quality of the reading, how many times the section of protein has been examined and found to be different (because each part of DNA is analysed multiple times), and whether the variant has been seen before.

To test the heterogeneous nature of the HDC system, we tested it with medical data. Two types of file were used for these use cases, both of which were text data, and are described in this section.

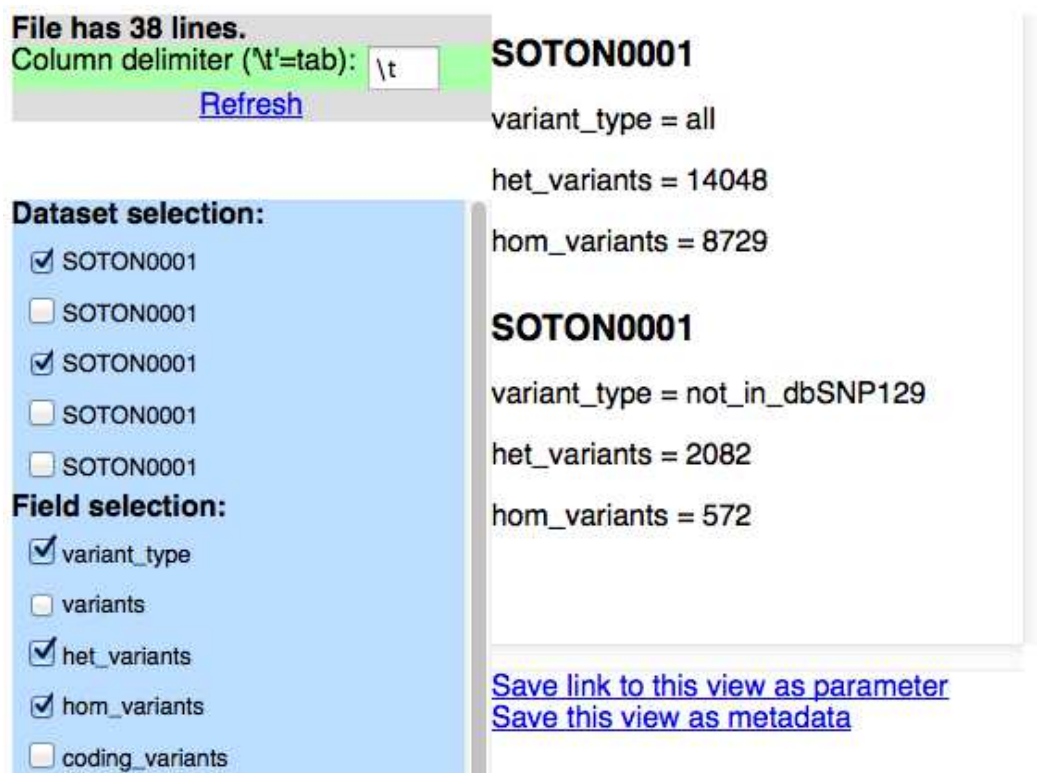
**Exome sequenced tab-delimited data:** The first type of file was the final output file for one patient sample having undergone exome sequencing and was mainly tab-delimited data, although some fields contained sub-fields. The variant data produced is in tabular form with one row per variant and includes:

- The location of the variant in the protein.
- The values of the reference nucleotide and the variant.
- The quality of the reading.
- How many times the section of protein has been examined and found to be different (because each part of DNA is analysed multiple times).
- Whether the variant has been seen before.

Once an exome-sequenced file is copied into the HDC share, the tab-delimited file plug-in can be activated from the HDC interface. The first row in the file gives the field names and the remaining rows give the data values for each field (which represents a variant). Specific columns or rows from the file can then be selected and the view can be exported to a new text file or saved as metadata in the data set. The plug-in in use is shown in Figure 5.4. The plug-in gives an overview of the data, but would have to be customised to support the sub-fields of the file format. The creation of a new plug-in in the next example shows the system is flexible enough to allow this.

**Summary metadata:** The second type of file tested was a `.stats` file containing summary data, with totals of variants for classifications of the data.

The file could be copied to the HDC and browsed using the standard delimited file browser, but the format of this file did not follow a simple tab-delimited file with a header row and value rows. In fact, the first column contained the field names and the remaining columns contained the values for each data set, with the first row of the file giving the name of the data set. This limited what could be done with the file in the HDC, and potentially valuable metadata would not be importable because the standard delimited file browser could not associate data values with field names (because data values for a field were in rows not columns).



**Figure 5.9:** Browsing the contents of a file containing summary data. The user can choose which data sets and fields they want to view, and can then import the data as metadata of the data set or just save a link to the report.

In this case, a plug-in was created to support this particular file format. The plug-in allows users to choose which fields and which data sets to view, then a link to the report can be saved or the selected data can be imported as metadata into the data set. Figure 5.9 shows this plug-in in use.

This demonstrates the flexibility and extensibility of the system. Without the additional plug-in, the data can still be uploaded, downloaded, given a description in the interface and discovered using searches, but the data-specific plug-in enables users to further interact with the data.

### 5.3.2 Finding data using SPARQL

Listing 5.1 shows how it is possible to use the relationships between data sets to find out which data sets have linked to a particular data set using SPARQL code. In this example, assuming we know that the data set with the identifier of 16 is a material, this lists which data sets have used that material in an experiment. Listing 5.2 expands upon this by filtering the results to only show the fatigue tests that have been performed on the material.

Listing 5.3 shows how a list of materials in the HDC can be retrieved while Listing 5.4 expands this by also filtering on the data set's metadata, showing materials that have a chromium content of greater than 10%. It finds a `Composition` property (in the `hdcparms` namespace) and retrieves the sub-parameter for chromium and then filters by its value, demonstrating how the nested metadata in the HDC can be queried.



**Listing 5.1:** Select all experiments that have have been performed on the material with the identifier of 16.

```
1 PREFIX dcterms: <http://purl.org/dc/terms/>
2 SELECT ?exp ?expname
3 WHERE {
4   ?material dcterms:identifier 16 .
5   ?material dcterms:references ?exp .
6   ?exp dcterms:title ?expname .
7 }
```

**Listing 5.2:** Select all fatigue tests that have been performed on the material with the identifier of 16.

```
1 PREFIX dcterms: <http://purl.org/dc/terms/>
2 SELECT ?exp ?expname
3 WHERE {
4   ?material dcterms:identifier 16 .
5   ?material dcterms:references ?exp .
6   ?exp dcterms:title ?expname .
7   ?exp dcterms:type ?exptype .
8   FILTER (?exptype = "Fatigue test")
9 }
```

**Listing 5.3:** Select all materials.

```
1 PREFIX dcterms: <http://purl.org/dc/terms/>
2 SELECT ?material ?materialname
3 WHERE {
4   ?material dcterms:title ?materialname .
5   ?material dcterms:type ?materialtype .
6   FILTER (?materialtype = "Material Data Sheet")
7 }
```

**Listing 5.4:** Select all materials with a chromium content of over 10.

```
1 PREFIX dcterms: <http://purl.org/dc/terms/>
2 PREFIX hdcparams: <https://hdc.soton.ac.uk/params/>
3 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
4 SELECT ?material ?materialname ?chromiumcontent
5 WHERE {
6   ?material dcterms:title ?materialname .
7   ?material dcterms:type ?materialtype .
8   ?material hdcparams:Composition ?comp .
9   ?comp hdcparams:Cr ?chromiumcontent .
10  FILTER ( ?materialtype = "Material Data Sheet"
11    && xsd:float(?chromiumcontent) > 10 )
12 }
```

### 5.3.3 EP2DC revisited

In Chapter 3 we showed how it was possible to adapt EPrints to store data sets in a data centre at the same time as the deposition of its associated paper, and maintain the link between them. We produced a web service layer that permitted EPrints to upload data and communicate with an early prototype of the Materials Data Centre. Since the prototype, the data centre architecture has been changed considerably, moving the data set storage out of Microsoft SharePoint and into a file system. However, several compatible design features still remain: data sets are used, the equivalent of the containers used in EP2DC; all data sets and data files still have IDs (no longer GUIDs); and the related data function still exists and is much improved so related data can still be returned.

To discover whether the service layer could be integrated into the MDC/HDC system we took the service layer designed for the EP2DC project (detailed in Table 3.1) and reimplemented the following operations:

**CreateContainer:** This creates a folder with a random name (not a GUID), waits ten seconds for the file system watcher to recognise the data set and returns the data set's new ID for EPrints to store.

**UploadFile:** This saves the file into a data set, waiting ten seconds for the file system watcher to recognise the data file and returning the new identifier.

**GetURIForMDCID:** This returns a string containing the location a file can be downloaded from.

**GetMDCIDsInContainer:** Given the identifier of a data set, this returns the IDs of all of the data files in a container.


**GetHTMLForMDCID:** This uses the more advanced recommendation functions, described in Section 4.2.3, to generate HTML to return to EPrints for embedding in its web page.

These operations were sufficient to demonstrate the basic interaction between EP2DC and the HDC. `AddURIToContainer`, `AddMDCIDToContainer`, `GetMDCIDForObjectPath` and XML validation were not implemented for this test. `AddURIToContainer` would simply add the URI as a metadata entry, `AddMDCIDToContainer` might create a link to a data set or add the URI as a metadata entry, `GetMDCIDForObjectPath` may not be required, and XML validation would be implemented using the same validation code given in Listing 3.2.

Figure 5.10 shows EP2DC using the HDC's implementation of the REST web service. The figure shows the improved data recommendations integrated into the EPrints record for a materials engineering article.

This has brought the evolution of the project full circle. The data centre as it matured has concentrated on the management of data during its preparation and use; the EP2DC plug-in demonstrates potential for the management of data at the end of its life cycle.

EP2DC




[Home](#) | [About](#) | [Browse by Year](#) | [Browse by Subject](#) | [Browse by Division](#) | [Browse by Author](#)

Logged in as Mr Mark Scott | [Manage deposits](#) | [Profile](#) | [Saved searches](#) | [Logout](#)

[Search](#)

### Life assessment methods for industrial steam and gas turbines blade to disc interfaces

Soady, K. and Reed, P.A.S. and Mellor, B.G. and Morris, A. (2009) *Life assessment methods for industrial steam and gas turbines blade to disc interfaces*. 15th Postgraduate Conference in Engineering Materials . pp. 5-6. (Unpublished)



[Microsoft Word](#)  
 3048Kb

**EP2DC - data management and collation for EPrints**

The following data is available to support this item:


Data available from: [Materials Data Centre](#)

[+ Show details](#)

[- Hide related items](#)

Suggested experiments based on existing links:

- [material\\_n18](#)
- [tensile\\_n18](#)
- [material\\_lshr\\_cg](#)
- [fatigue\\_n18](#)
- [material\\_n18](#)



Experiments of the same type:

- [tensile\\_n18](#)
- [tensile\\_lshr\\_cg](#)
- [tensile](#)

Data files of the same type:

- [mdc\\_material\\_n18\\_microstructure\\_a.png](#)
- [material\\_lshr\\_cg\\_microstructure.png](#)
- [head-bone-detail-flut.png](#)

<b>Item Type:</b>	Article
<b>Subjects:</b>	<a href="#">T Technology &gt; TA Engineering (General)</a> , <a href="#">Civil engineering (General)</a>
<b>Divisions:</b>	<a href="#">Faculty of Engineering, Science and Mathematics &gt; School of Engineering Sciences</a>
<b>ID Code:</b>	5
<b>Deposited By:</b>	Mr Mark Scott
<b>Deposited On:</b>	13 Nov 2011 11:15
<b>Last Modified:</b>	13 Nov 2011 11:15

Repository Staff Only: [item control page](#)

Test Repository is powered by [EPrints 3](#) which is developed by the [School of Electronics and Computer Science](#) at the University of Southampton. [More information and software credits](#).




Figure 5.10: EP2DC with the new HDC model.

## 5.4 Discussion

This section evaluates the features of the HDC system with respect to the following aims discussed in the introduction: a straightforward data deposit process; data set tools adding value to users to encourage uptake; data set discoverability; and metadata and metadata tools. We will also discuss cross-discipline support in Section 5.4.5.

### 5.4.1 Straightforward data deposit process

Keeping the data deposit as simple as possible has advantages in that users on the system have not been deterred by a lengthy deposit process. It also allows submission from sources that are traditionally unable to interact with the repository, such as laboratory equipment; metadata can be collected using automated metadata parsing.

As a general guideline when depositing in the system, users should try to provide a description for the data set and images to improve reports, name their files appropriately to help tools such as automatic data import or the CT data viewer (shown in Section 4.3.4), set the correct data set type to gain access to metadata templates, and use data set relationships to improve recommendations and search results (such as the SPARQL examples shown in Section 5.3.2).

### 5.4.2 Metadata and metadata tools

The provision of a sortable and hierarchical metadata system allows the framework to support many metadata needs, as we showed in the use cases in Sections 4.3 and 5.3. It can be employed by users to record their own metadata (such as the tensile test, fatigue test and material information sheet use cases), by tools to save data to support their functionality (such as saving the data set's thumbnail), and empty data sets – where only the metadata features are used – can be useful to create reports and templates. We have been able to provide templates to users and reports that match the users' requirements as demonstrated with the reports in Section 4.3 and templates in the materials information sheet use case in Section 4.3.1.

Because the interface is less strict than a deposit system might be, metadata that is mandated by a data archival system might not be captured at this stage. This is mitigated to a certain extent by the fact that not all data sets may reach the archival stage (because the system is aimed at users whilst they are still using their data), any metadata that is obtained can be readily transferred to a data archive, by the provision of tools useful to the user that rely on accurate metadata and by warning users who have ignored compulsory values or entered incorrect data types. If the metadata was to be used at the curation stage, a manual step before publication may be required.

Providing tools that require metadata can help capture the correct information. For example, some of the features of the HDC rely on data set metadata such as the data set recommendations feature introduced in Section 4.2.3, whilst other features can be helped with metadata such as reporting. Features such as templates and metadata copying help to make metadata entry easier.

A potential omission in the design of the system is the lack of metadata per data file. The researcher interviewed for the human genetics case study in Chapter 6 reported that data files were sometimes stored by the thousand in a single folder and then processed using scripts; for this, metadata per file could be invaluable and the data set-centric approach could be a hindrance. Using the nesting capabilities of the existing system, data file metadata can in fact be stored, using a parent parameter for each file. The API could be adapted to take this into account and provide metadata per file (even though the metadata would be stored per data set). Unrelated data files could then be stored in a single data set and the scripts would still function with the metadata database providing support to the scripts. If this were to prove insufficient, the database could be modified with the addition of three tables (`DataFileParameters`, `DataFileParameterLinks`, and `DataFileTypeDataFileTemplateMap`), mirroring the approach of data set metadata and permitting nesting and templating.

### **5.4.3 Data set discoverability**

The system aims to encourage sharing – as well as make it easier to discover data sets – with a number of features: data set relationships provide ways of organising data sets (e.g. linking and collections); data set search helps to find data by data set name, file name or parameter name; and data set recommendations helps to reveal data sets that might be of interest. The SPARQL end-point permits more advanced querying of the HDC, for example to find data sets with specific features.

The ability to use tools like a SPARQL client against the data supplements the system's reporting and searching capabilities, and permits scenarios such as a script that can query the data centre to find data sets with certain properties for use in a simulation. We have shown examples of SPARQL queries designed with the help of users in Section 5.3.2, such as listing experiments made with a particular material or finding materials with a particular composition.

The ability to easily discover data sets based on their relationships has important implications in a discipline such as materials engineering. For example, if it is ever discovered that a piece of equipment was defective, retrieving all data sets that used that equipment (perhaps after a specific date) would be simple if the user had created a data set containing apparatus metadata with calibration data and maintenance schedule information, and linked to it. Other data that could be stored about the apparatus includes its manufacturer, model and serial numbers, features (such as maximum load), specifications and dimensions, photographs of the equipment, manuals and schematics.

For these examples to work at their best, users have to ensure they provide the system with appropriate metadata such as setting the type of data set, providing metadata and using relationships. Without this information in the system, users might find it more difficult to discover new data sets or find their own.

#### 5.4.4 Data set tools adding value to users

Many users may never go any further than copying data into the HDC's network share, which would be a perfectly valid use of the system for those just wanting to take advantage of the system as somewhere to store or backup their data and could enable others to annotate their data too (e.g. a curator or another user); however, we created tools to permit users to perform additional work with the data that they have uploaded. This includes reporting tools, security editing to allow sharing of data, a CT data file browser, a thumbnail for the data set and a tab delimited file browser. We showed the use of some of these tools in Section 4.3 and Section 5.3.

The ability to produce reports containing data and metadata from multiple data sets allows the user to take advantage of the relationships that exist naturally between data sets. Producing a report presenting data about a fatigue test is made easier as the user is able to pull in metadata about the material into the report as well as the results of the fatigue test.

These features give users extra control over their data, providing an additional incentive to upload into the system. For example, the ability to edit the security of a data set enables users to easily share data sets with their colleagues without relying on an administrator.

Some of the tools are generic in nature whilst others have been written specifically for a particular type of data, such as the medical data in Section 5.3. This illustrates the flexibility and extensibility of the system but also shows that there is work involved to make the best use of the system and customise it for its users.

Some of the tools, such as the graphing and reporting plug-ins, cannot replace tools such as Microsoft Excel when it comes to analysing, for example, tensile test data. Likewise, the CT image data browser plug-in cannot compete in performance or features against specialised software for processing such large data sets. What the tools provide are overviews of the data and the ability for comparing data across data sets. This alone might encourage users to deposit their raw data so – in the charting plug-in's case – they could produce scatter plots comparing against other data sets immediately. The raw data can then be imported into Microsoft Excel for further analysis which the user may then choose to keep with the raw data in the HDC.

#### 5.4.5 Cross-discipline support

The system we have presented has been used for storage of materials engineering data and a limited amount of human genetics data. For each of these disciplines, custom tools were written; for example, metadata import tools, data browsing tools and custom reports. We have been able to rapidly customise and create custom tools for discipline specific data as we demonstrated with the medical summary data file in Section 5.3.

While designed for materials engineers, the CT image data browser shown in Section 4.3.4, data set reporting, data set thumbnail and graphing plug-ins might all be useful for data from other disciplines. The delimited file viewer plug-in can be used for browsing data from any discipline that follows a delimited file format. If plug-ins have not been created for a specific file format, the files are still supported by the underlying file system and can still be given a description and downloaded. The aim of the plug-ins is to encourage users to supply correct metadata and add value to the user when they choose to upload their data into the repository.

### **5.4.6 Future work**

In this section we have evaluated the system against the aims we set out in the introduction. The system itself is still young and only has a small number of users so our conclusions are based on their initial feedback. What we have shown is that the framework of a file system, file system monitor, metadata database and an extensible set of interface tools is flexible and can be used to capture and manage heterogeneous data and metadata.

It was mentioned in Section 2.5.3 that MatDB data files could be stored in the HDC without validation, but data uploaded to the HDC via the EP2DC web service can be validated before it is stored. HDC data file plug-ins provide a way of validating data files in the data store and, for MatDB data files, the plug-in could call the EP2DC validation routine to permit users to validate their MatDB data before it is published.

## **5.5 Summary**

In this chapter, we presented plug-ins that permit materials engineers to work with data files in the system with some allowing comparison of data across data sets, and we demonstrated automatic metadata ingestion. We also discussed additions to the model in Chapter 4 that were required to turn it into a usable system for the storage of in-use, pre-publication heterogeneous data, such as security, configuration of the monitoring service and support of RDF.

The system we presented in Chapter 4 and developed further in this chapter provides a simple data deposit process, requiring a user only to create a folder in a network share to create a data set in the HDC system. We have shown this through a number of use cases in Section 4.3 and Section 5.3. The aim of this is to capture user data earlier in its life cycle while the data is still in use.

Additional methods of depositing data sets are also provided, such as through the web interface, shown in Section 5.2.7, and even EPrints with the EP2DC extension which we first showed in Chapter 3 and revisited in Section 5.3.3. Metadata entry is entirely optional, but encouraged to get the best of the system. Tools which rely on the metadata being present can help motivate users to provide it as we illustrated with the material information use case reports in Section 4.3.1 and SPARQL query examples in Section 5.3.2.

Removing any mandatory metadata requirements reduces the effort and time required to submit data to the repository. Metadata can then be obtained by other means, such as automated extraction from a data file with a plug-in, which we discussed with the `.xtekt` metadata accompanying CT data in Section 5.2.6.

Relationships between data sets enhance the user's experience and providing tools for users to query, analyse and report on their data sets increases the flexibility of the system. Our framework has also proven to allow customisation as shown with the summary data file in the medical data case study in Section 5.3.

Using a file system as the place where users upload their data sets allows heterogeneous data to be supported in a research group repository, and we have demonstrated this with a number of data sets, including large computed tomography data in Section 4.3.4 and small

textual data files in Section 5.3. It also permits collection of data sets from equipment which can also deposit data into the file share; any metadata generated during the test by the apparatus can be automatically imported, as discussed in Section 5.2.6.

By providing multiple entry points to the system and a solid set of metadata tools, standards can still be supported while keeping the data deposit process simple for the user. As our user base increases we expect to discover more plug-ins are required, improvements can be made to the interface and additional reports provided, while some of the existing tools may need improving for performance or usability reasons. However, this is all ornamentation to the foundation of data in the file system, and sortable and hierarchical metadata in a database.

The next chapter presents one of the first attempts at the University of Southampton at teaching data management concepts to early career researchers, showing the similarities and differences between data in different disciplines, its importance, and giving some tips to users on how it can be managed.





## Chapter 6

# Research data management education

---

**The work in this chapter was used to educate users about the concepts of research data management at the University of Southampton. It has been presented as:**

Mark Scott, Richard P. Boardman, Philippa A. Reed, Dorothy Byatt and Simon J. Cox. 2013. 'Research data management education for future curators'. In *8th International Digital Curation Conference*. Amsterdam, Netherlands: The Digital Curation Centre, 14th January

Wendy White, Simon Cox, Graeme Earl, Mark Scott, Dorothy Byatt and Steve Hitchcock. 2013. 'Working collaboratively with PhD and early career researchers: Agents for change'. In *8th International Digital Curation Conference*. Amsterdam, Netherlands: The Digital Curation Centre, 14th January–17th January

**and selected to be published as:**

Mark Scott, Richard P. Boardman, Philippa A. Reed and Simon J. Cox. 2013b. 'Research data management education for future curators'. *The International Journal of Digital Curation* 8 (1): 288–94. ISSN: 1746-8256. doi:10.2218/ijdc.v8i1.261

**The guide presented in this chapter is available at:**

Mark Scott, Richard P. Boardman, Philippa A. Reed and Simon J. Cox. 2013a. 'Introducing research data'. Edited by Mark Scott and Dorothy Byatt. JISC DataPool Project, 14th March. Accessed 13th May 2015. <http://eprints.soton.ac.uk/360442/>

---

## 6.1 Introduction



WE HAVE LOOKED at managing data and metadata whilst it is still in use with the MDC and HDC projects, and introduced data management at the end of its life cycle with the EP2DC project. However, building tools for data management does not guarantee its use.

During the course of the HDC project, and working with the engineering and medical users and their data, it became clear that there was a need to provide education about research data management. The creators of data need to understand their data, comprehend its importance and start to think about how they can start to look after it and ensure its preservation

for the generations to come. There were indeed many overlaps between disciplines – but also some differences – and there was a desire to provide some best practices on managing data.

We looked at five researchers' work from medicine, materials engineering, aerodynamics, chemistry and archaeology, and produced case studies showing the similarities and differences between the data types they produce. A guide, aimed at postgraduates, was created containing the case studies and an introduction to research data management. The concepts in the guide were also presented as training lectures.

In this chapter, we discuss how the guide was put together, provide the contents of the guide and present our findings.

## 6.2 About the *Introducing Research Data* guide

The case studies were written up into a glossy, introductory guide which was broken down into three parts: Part I, 'Five Ways To Think About Research Data', introduced research data, its categorisation and the data life cycle; Part II, 'Case Studies', provided five case studies from medicine, materials engineering, aerodynamics, chemistry and archaeology; and Part III, 'Data Management Practices', included general advice on managing data. This section gives more information about each part.

### 6.2.1 Guide Part I: 'Five Ways To Think About Research Data'

Combining some recognised definitions, this part of the guide introduces research data using the perspectives below, which helped to set the scene and introduce types of data and the research that data represented. The case studies in Part II were written using this framework.

**'How research data is collected':** Introduces where data comes from, using categories defined by Research Information Network (2008).

**'The forms of research':** Lists items of research that the data represents to provide perspective, adapted from University of Edinburgh (2011).

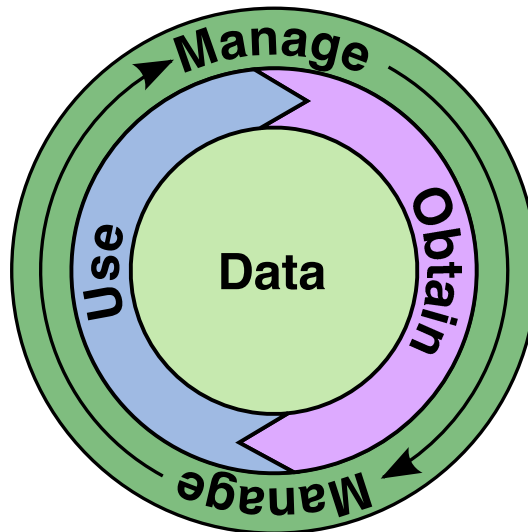
**'Electronic storage of the research data':** Discusses how data is stored on a computer, adapted from University of Edinburgh (2011).

**'Electronic data volumes':** Gives an overview of the sizes and structures of the files and data sets.

**'Data life cycle':** Provides four ways of representing the stages data goes through during its lifetime.

### 6.2.2 Guide Part II: 'Case Studies'

The aim of the case studies was to show the similarities and differences in research data between disciplines and how this was managed. A researcher from each discipline was interviewed about the data that was used in a particular example in their field. Each case study was written up using terminology from the discipline, whilst remaining accessible to non-experts.



**Figure 6.1:** A simplified data life cycle, used as section headings for the case studies.

As can be seen in Section 6.3 (for example, Figures 6.12–6.14), the format of the guide was presented clearly and used colour extensively to make it more approachable and easier to digest. Each case study begins with a table summarising the data categories used by the researcher, grouped using the framework introduced in the previous section. It then includes a discussion of the researcher’s practices when producing and using the data, broken down into three sections using steps taken from one of the data life cycles in Part I of the guide (shown in Figure 6.1): ‘Obtaining the data’, ‘Using the data’ and ‘Managing the data’. Finally, each researcher was asked to provide some images that showed their research or data in use and the case study was formatted to fit within four pages.

### 6.2.3 Guide Part III: ‘Data Management Practices’

The final part of the guide provided general advice on how to manage data. Topics include file naming, data preservation, file tracking, file formats, backups and file versioning.

The part was broken into three sections:

**‘Storing data’:** Discussing file naming, data preservation, file formats and folder structuring.

**‘Backups’:** Gives a number of suggestions on how data might be backed up.

**‘Version management’:** Discusses methods of tracking versions of files for researchers who need to manage large amounts of files that are modified often.

### 6.3 The *Introducing Research Data* guide

The previous section introduced the structure of the guide and how it was created. This section presents the contents of the guide in full in Figures 6.2–6.32 on the following pages.

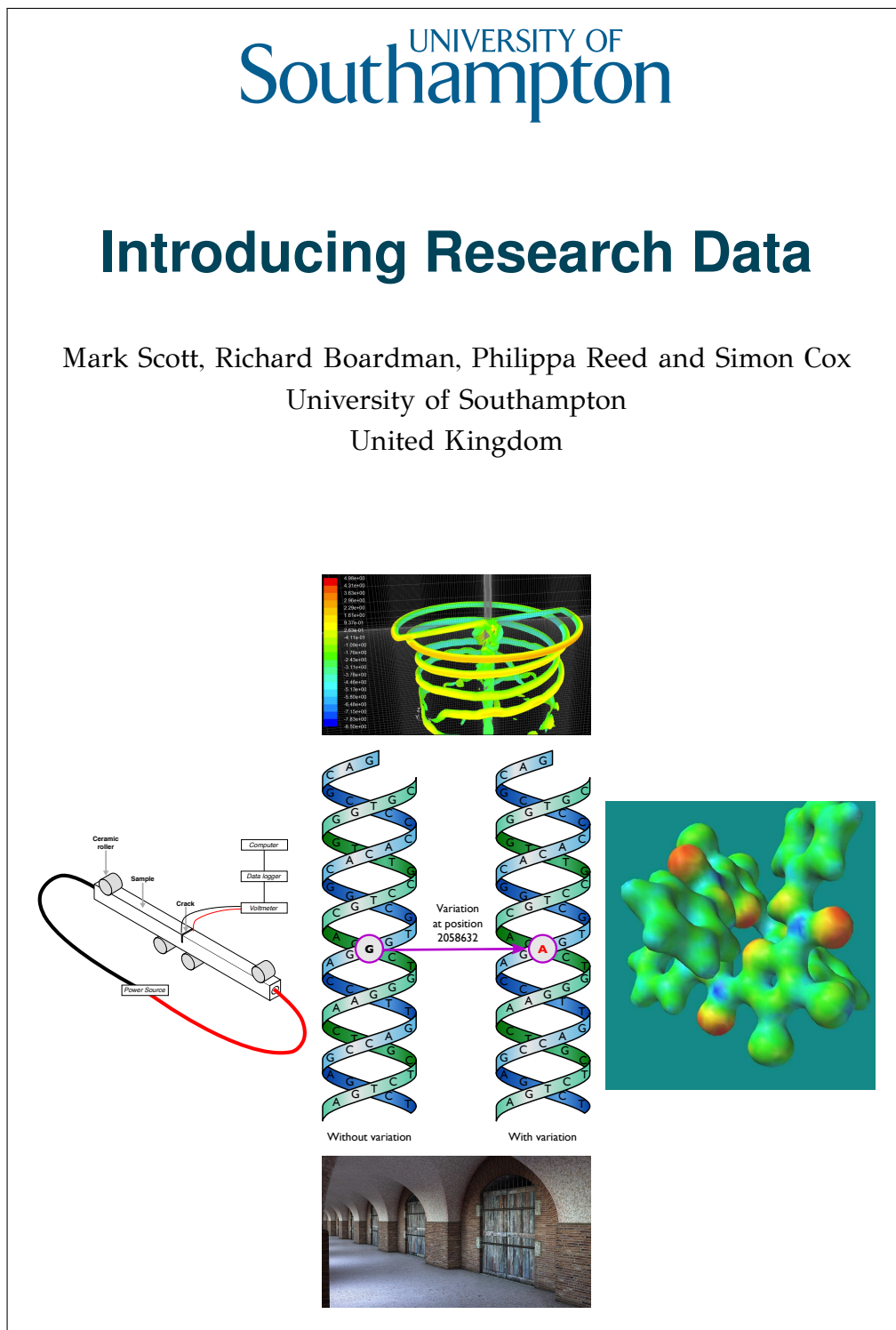


Figure 6.2: The *Introducing Research Data* guide front page.

## Introducing Research Data

© University of Southampton 2013

Overall content by Mark Scott, Richard Boardman, Philippa Reed and Simon Cox in the Faculty of Engineering and the Environment.

University edition revised by Mark Scott and Dorothy Byatt.

Case studies produced with help from Andy Collins, Thomas Mbuya, Kath Soady, Gregory Jasion, Simon Coles and Graeme Earl.

This is revision 115:1e83a78ded3a of this document, created 2013-03-14 12:50.

Figure 6.3: The *Introducing Research Data* guide copyright page (page i).

## Introducing Research Data

Every discipline, from the arts and humanities to physics, is increasingly using data to drive forward its goals. Medicine might use it for recording the statistics of a particular drugs trial; physicists have complex experiments—such as the the Large Hadron Collider at CERN—producing massive quantities of data on an hourly basis; and archaeologists meticulously preserve digital records of excavation sites and artefacts.

Data comes in many forms: small, large, simple, complex and colourful. This guide first introduces the forms data can take by showing five ways of looking at data, then presents some case studies of data usage in several disciplines in an attempt to illustrate the types of data you might encounter in your research and give you some tips or tricks that will help you in your own discipline.

The final part of the guide gives some general advice on managing and understanding data.

Figure 6.4: The *Introducing Research Data* guide abstract page (page ii).

Table of Contents	
<b>I Five Ways To Think About Research Data</b>	<b>1</b>
1 How research data is collected	1
2 The forms of research	2
3 Electronic storage of the research data	3
4 Electronic Data Volumes	3
5 Data life cycle	4
<b>II Case Studies</b>	<b>6</b>
1 Medicine: Human Genetics	7
2 Materials Engineering: Long crack growth fatigue testing	10
3 Aerodynamics: Simulations Using Computational Fluid Dynamics	13
4 Chemistry: Crystal Structures	16
5 Archaeology: An Excavation	19
<b>III Data Management Practices</b>	<b>23</b>
1 Storing data	23
2 Backups	24
3 Version management	26
<b>IV Acknowledgements</b>	<b>27</b>

Figure 6.5: The *Introducing Research Data* guide table of contents page (page iii).

**Part I****Five Ways To Think About Research Data**

Science has progressed by “standing on the shoulders of giants” and for centuries research and knowledge has been shared through the publication and dissemination of books, papers and scholarly communications. Moving forward much of our understanding builds on (large scale) data sets which have been collected or generated as part of this scientific process of discovery. How will this be made available for future generations? How will we ensure that, once collected or generated, others can stand on the shoulders of the data we produce?

Deciding on how to look after data depends on what your data looks like and what needs to be done with it. You should find out if your discipline already has standard practices and use them. We hope that this brief introduction will give some templates of what is already being done in a few disciplines and enable you to start thinking about what you might do with your research data to make it accessible to others.

This part of the guide introduces five ways of looking at research data.

**1 How research data is collected**

The first way of thinking about research data is where it comes from (Research Information Network, 2008). Each of the case studies in part II illustrates one of these categories.

- Scientific experiments, *e.g.* materials engineering fatigue test in section 2
- Models or simulations, *e.g.* CFD rotor wake simulation in section 3
- Observations (of a specific, possibly unrepeatable, event at a specific time or location), *e.g.* an archaeological dig in section 5
- Derived data (taking other data and performing some processing it), *e.g.* chemical structures in chemistry in section 4
- Reference data, *e.g.* The reference human genome sequence in section 1

**Figure 6.6:** The *Introducing Research Data* guide (page 1).



## 2 The forms of research

Research can come in many different forms, some electronic and some physical. The following list gives an idea of these different forms.

- Electronic text documents, *e.g.* text, PDF, Microsoft Word files
- Spreadsheets
- Laboratory notebooks, field notebooks and diaries
- Questionnaires, transcripts and codebooks
- Audiotapes and videotapes
- Photographs and films
- Examination results
- Specimens, samples, artefacts and slides
- Digital objects, *e.g.* figures, videos
- Database schemas
- Database contents
- Models, algorithms and scripts
- Software configuration, *e.g.* case files
- Software pre-process files, *e.g.* geometry, mesh
- Software post-process files, *e.g.* plots, comma-separated value data (CSV)
- Methodologies, workflows, standard operating procedures and protocols
- Experimental results
- Metadata (data describing data), *e.g.* environmental conditions during experiment
- Other data files, *e.g.* literature review records, email archives

Figure 6.7: The *Introducing Research Data* guide (page 2).

### 3 Electronic storage of the research data

The third way of thinking about research data is how it is stored on a computer. Here are some of the categories of electronic data:

- Textual, *e.g.* flat text files, Microsoft Word, PDF, RTF
- Numerical, *e.g.* Excel, CSV
- Multimedia, *e.g.* image (JPEG, TIFF, DICOM), movie (MPEG, AVI), audio (MP3, WAV, OGG)
- Structured, *e.g.* multi-purpose (XML), relational (MySQL database)
- Software code, *e.g.* Java, C
- Software specific, *e.g.* mesh, geometry, 3D CAD, statistical model
- Discipline specific, *e.g.* Flexible Image Transport System (FITS) in astronomy, Crystallographic Information File (CIF) in chemistry
- Instrument specific, *e.g.* Olympus Confocal Microscope Data Format, Carl Zeiss Digital Microscopic Image Format (ZVI)

### 4 Electronic Data Volumes

Another consideration when evaluating research data is the size of the files. These are subjective, *e.g.* a set of photographs may be considered to be large to that researcher, but another researcher may work with three dimensional X-ray data which can be many times larger.

- Individual large file, *e.g.* database; virtual machine's hard disk; raw CT data; movie
- Set of small files, collectively large, *e.g.* time steps in CFD simulation (collectively representing the full simulation, but individually a subset of time which can be processed separately); individual frames of movie
- Set of small files, collectively small, *e.g.* source code where the entire set of files are needed to compile
- Individual small file, *e.g.* CSV files produced by a numerical solver; photograph
- Combinations of the above, *e.g.* a large file accompanied by a small text file describing its contents.

Figure 6.8: The *Introducing Research Data* guide (page 3).

5 Data life cycle

During its lifetime, data goes through a number of phases. Different disciplines have different ways of thinking about this life cycle as can be seen by the figures in this section.

The first two figures show a high-level way of considering this process as a life cycle, whereas the second two present it more as a series of discrete steps in a workflow. Figure 1 illustrates a simplified view of data from a researcher's point of view and figure 2 is the life cycle from a curator's perspective. The steps in figure 4 are applicable to a researcher who performs processing on their data and figure 3 is another variation (Humphrey, 2006).

Figure 1: Data life cycle as a user/creator of data

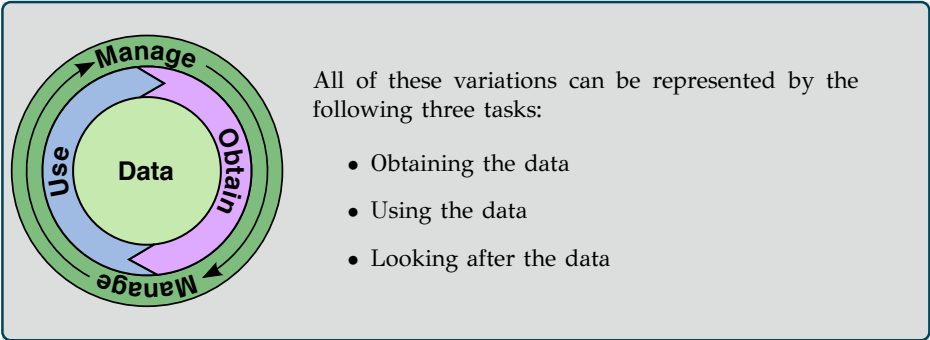


Figure 2: Data life cycle from the perspective of a curator (Digital Curation Centre, 2010)

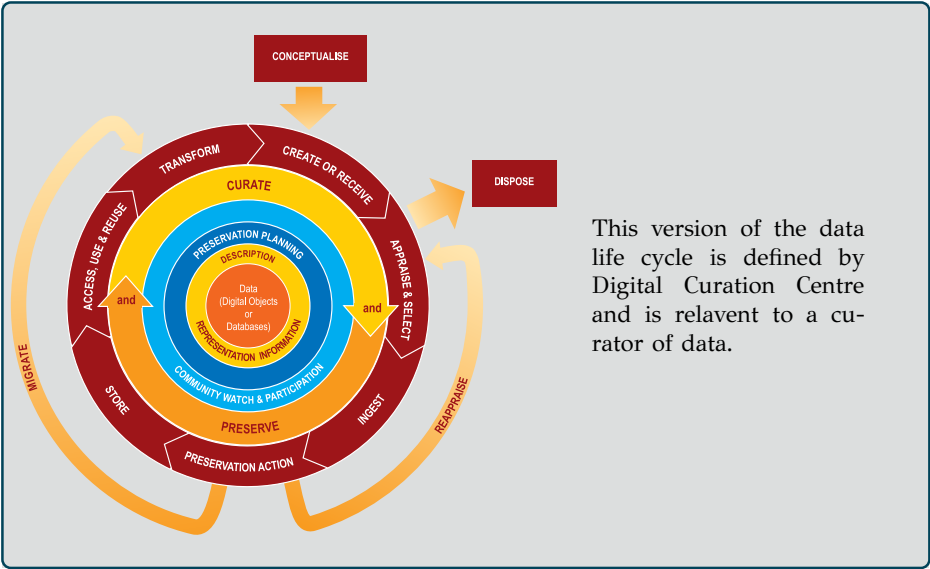


Figure 6.9: The *Introducing Research Data* guide (page 4).

Figure 3: Stages in data life cycle at a research project level (Humphrey, 2006)

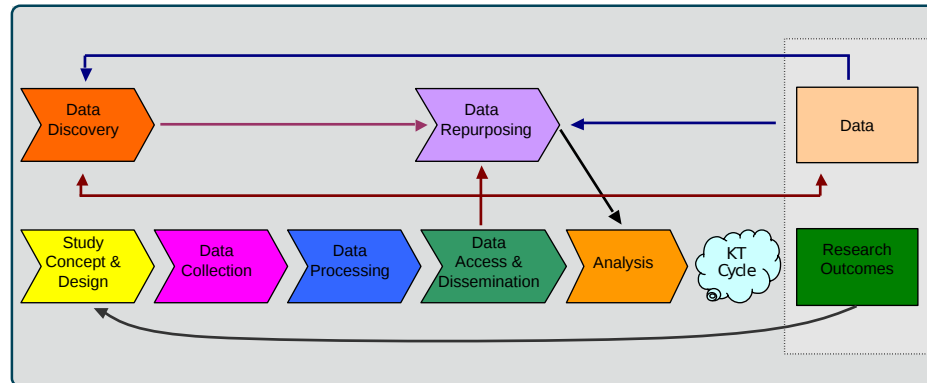
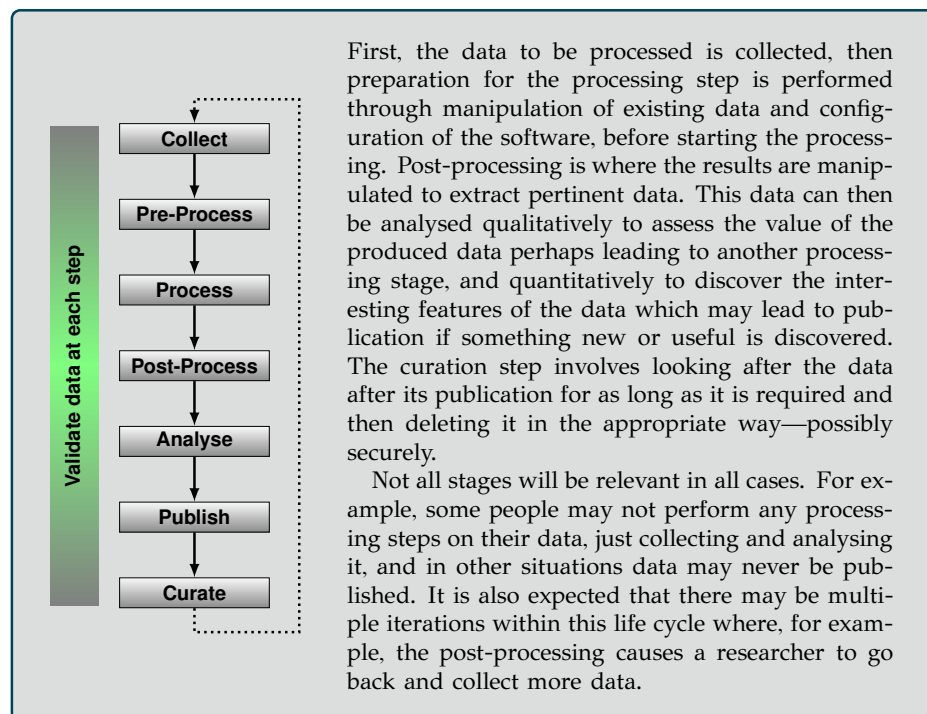


Figure 4: Stages in data life cycle as a user/creator of data

Figure 6.10: The *Introducing Research Data* guide (page 5).

## Part II

### Case Studies

The table below illustrates how the categories introduced in part I are used in the case studies that follow. It is not comprehensive as all researchers have their own methods.

	Human Genetics	Fatigue Test	CFD	Crystallography	Archaeology
● Case study provides good example					
○ Also relevant in case study					
<b>Sources of data</b>					
Scientific experiments	○	●		○	
Models or simulations	○		●		
Observations		○			●
Derived data	○	○		●	
Reference data	●	○			○
<b>Forms of research</b>					
Electronic text documents	○		●	○	●
Spreadsheets	○	●			●
Laboratory notebooks, diaries					●
Questionnaires, transcripts, codebooks					
Audiotapes, videotapes					●
Photographs, films					●
Examination results					
Specimens, samples, artefacts, slides	●			●	●
Digital objects	○	○	●	○	●
Database schemas					○
Database contents					●
Models, algorithms, scripts	○	○	●	○	
Software configuration			●		
Software pre-process files	○	○	●	○	
Software post-process files	○	○	●	○	
Methodologies, workflows, procedures					●
Experimental results				●	
Metadata					●
Other data files					
<b>Electronic representation of data</b>					
Textual	●		●		●
Numerical	○	●			●
Multimedia		○	○	○	●
Structured	●			●	●
Software code			○		
Software specific			●	○	●
Discipline specific	●			○	●
Instrument specific				●	●

Figure 6.11: The *Introducing Research Data* guide (page 6).

## 1 Medicine: Human Genetics

This chapter discusses an example of data produced in medical research. This provides a good illustration of using reference data in research, in that the sample data is compared against a reference data set.

### Data categories in this case study

- Case study provides good example
- Also relevant in case study

#### Sources of data

Scientific experiments	○ Analysis of gene sequence
Models or simulations	○ Genome Analysis Toolkit processing
Derived data	○ Aligned data
Reference data	● Human genome reference sequence

#### Forms of research

Electronic text documents	○ Journal publication containing details of discoveries
Spreadsheets	○ For gene sequence analysis
Specimens, samples, artefacts, slides	● The DNA sequence
Digital objects	○ Gene sequence figures
Models, algorithms, scripts	○ Genome Analysis Toolkit files
Software pre-process files	○ FASTQ file
Software post-process files	○ Novoalign output

#### Electronic representation of data

Textual	○ Journal publication containing details of discoveries
Numerical	○ Spreadsheets containing analysis
Structured	● FASTQ files (structured, text-based format)
Discipline specific	● FASTQ files

### Data life cycle steps in this case study

#### Data life cycle stages

Collect	● Analysis of DNA sequence using sequencing machine
Pre-Process	● Align sequences against reference genome sequence with <i>Novoalign</i>
Process	● Process data with <i>Genome Analysis Toolkit</i>
Post-Process	● Filter results with <i>SIFT</i>
Analyse	● Analyse results with Microsoft Excel
Publish	● Discovery of genetic cause of a disease to a journal
Curate	● Upload sequence data to public genome databases

Figure 6.12: The *Introducing Research Data* guide (page 7).

## Background

### Obtaining the data

Researchers into human genetics take a DNA sample and analyse it using a sequencing machine which produces short sequence *reads* representing the sequence as millions of fragments. The fragments represent the sequence of 3 billion nucleotides producing a dataset of up to 50 GB. The most cost-effective strategy at this time is to only sequence the protein coding regions (exome) which is about 1% of this data. The data generated is in a text-based format known as *FASTQ*.

### Using the data

The FASTQ data is pre-processed by a software package called *Novoalign* to align the short reads against a complete human genome reference sequence. The aligned data can then be processed using the software tools in the *Genome Analysis Toolkit* to identify regions that are different from the reference data set.

Comparing sequences from two subjects' samples will identify thousands of differences, the majority of which make no difference to the protein that the gene codes for. *Synonymous* differences do not change the protein, but *non-synonymous* differences will. A DNA sequence with a *frameshift mutation* codes for a protein that is considered to be non-functional. However, an average adult will have a large number of these differences and still be healthy and research has only just begun to understand the variants and how they can cause diseases.

Filtering against genome databases permits known diseases to be identified by comparing the variants identified with those that are known causes of disease, and the effect a variant may have on the protein can be calculated by tools such as *SIFT* (*Sorting Intolerant From Tolerant*).

The variant data produced is in tabular form with one row per variant and includes:

- The location of the variant in the protein
- The values of the reference nucleotide and the variant
- The quality of the reading
- How many times the section of protein has been examined and found to be different (because each part of DNA is analysed multiple times)
- Whether the variant has been seen before

Microsoft Excel is frequently used for analysis of the variant data.

### Looking after the data

Data is stored in a number of formats as it is converted from type to type for each stage. New discoveries are fed back to the community by uploading to sequence databases so tools like SIFT can take advantage.

Figure 6.13: The *Introducing Research Data* guide (page 8).

Figure 5: Viewing genetic sequence mutation data (Broad Institute, 2012)

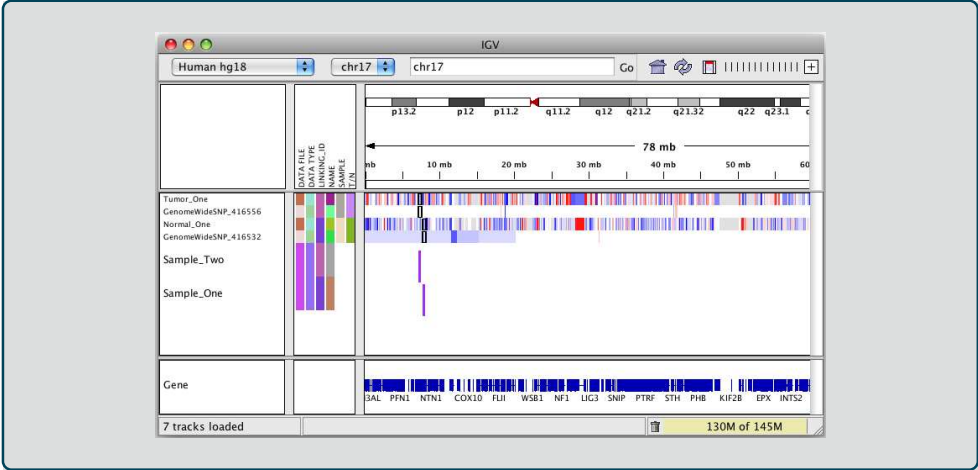
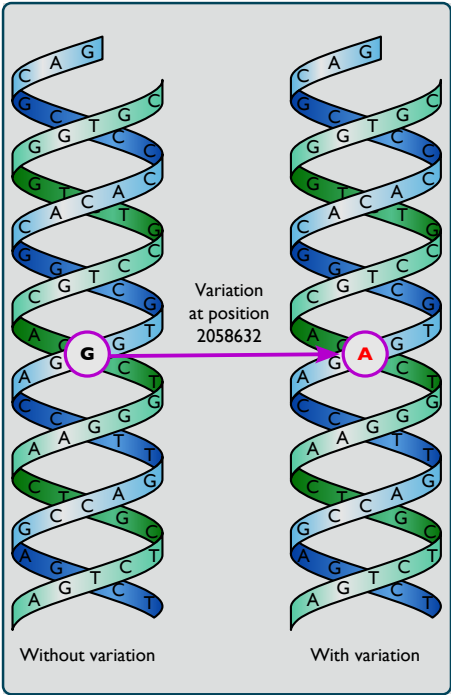


Figure 6: Variation in the *TSC2* gene, identified as one possible cause for *tuberous sclerosis*, a genetic disorder that causes tumors in organs around the body



### Summary

In this case study, the researcher collects DNA sequence data and, after preparing the data by aligning it, processes it to find regions that differ from a reference data set. Analysis of the data can involve comparing it against genome databases and manipulation in spreadsheet software. Publication will occur when patterns that relate to diseases are identified, which will then help improve the existing genome databases and analysis tools. This case study provided good examples of a wide range of research data including:

- Using reference data (the complete genome reference sequence, and the genome databases);
- Discipline specific data (the FASTQ files containing the DNA sequence reads).

Figure 6.14: The *Introducing Research Data* guide (page 9).



## 2 Materials Engineering: Long crack growth fatigue testing

This section will be used to give an example of data produced in materials engineering. This gives a good illustration of acquiring data through scientific experiment and then performing some processing on it.

### Data categories in this case study

- Case study provides good example
- Also relevant in case study

#### Sources of data

Scientific experiments	● The fatigue test
Observations	○ Monitoring PD values to find threshold
Derived data	○ Smoothed PD readings in Excel
Reference data	○ $K$ values for specimen geometry and load combination; PD calibration values for the material.

#### Forms of research

Spreadsheets	● Manipulated PD readings in Excel
Digital objects	○ Graph of $da/dN$ versus $\Delta K$ ; Images of the material's microstructure.
Models, algorithms, scripts	○ Smoothing algorithm
Software pre-process files	○ Raw CSV data of PD readings
Software post-process files	○ Manipulated PD readings in Excel

#### Electronic representation of data

Numerical	● CSV and Excel files
Multimedia	○ Graph of $da/dN$ versus $\Delta K$ (vector file); Material's microstructure (bitmap file).

### Data life cycle steps in this case study

#### Data life cycle stages

Collect	● PD readings from fatigue test
Pre-Process	
Process	
Post-Process	● Manipulating/smoothing PD readings
Analyse	● Plotting of stress against number of cycles to failure (S-N curve)
Publish	● Journal paper containing properties of a material
Curate	● Upload data to materials data repository

Figure 6.15: The *Introducing Research Data* guide (page 10).

## Background

### Obtaining the data

Starting with a sample of material that contains a crack, the growth of the crack is monitored as loads are applied to it. An electric current is passed through the sample and fluctuations in resistance or impedance—depending on whether direct or alternating current is being used—are detected by measuring the change in potential difference (PD) across the sample. PD increases as the resistance or impedance of the material increases, caused by the growth of the crack.

$K$  is a theoretical value representing the stress intensity of a crack based on the applied load and the crack's size and geometry. The range of  $K$  values to be applied to a sample in a fatigue test, known as  $\Delta K$ , is calculated in advance and load is adjusted to match the target  $\Delta K$  as the length of the crack changes.  $K$  values for a specimen geometry and load combination are usually pre-existing, having previously been calculated. If they are not available, finite element analysis methods are used to calculate them.

### Using the data

The raw data generated from a fatigue test consists of the PD readings in relation to time. The length of the crack ( $a$ ) is calculated from the PD voltage data and the number of loading cycles ( $N$ ) is known from the time elapsed. The rate at which the crack grows with respect to the number of loading cycles can then be calculated ( $da/dN$ ) and compared to the range of stress intensity ( $\Delta K$ ) to give a measurement of crack driving force and the material's ability to resist crack propagation at different stress intensities.

Calculating the length of the crack from a given PD value is possible because of previous calibrations with the material and specimen geometry. When using a material and specimen geometry where no data is available, early experiments do not necessarily yield any useful data other than data that can be used for PD calibration. It is helpful to find someone else who has already performed fatigue tests on a particular material and can give starting values for calculating crack length.

The captured data is often quite noisy so it may be necessary to smooth it during analysis. There are many methods for this, including skipping obviously erroneous sampled values or using a best-fit curve.

### Looking after the data

Raw data is stored as CSV and its manipulation is usually done using spreadsheet software such as Microsoft Excel.

Figure 6.16: The *Introducing Research Data* guide (page 11).

Figure 7: Fatigue Test Configuration

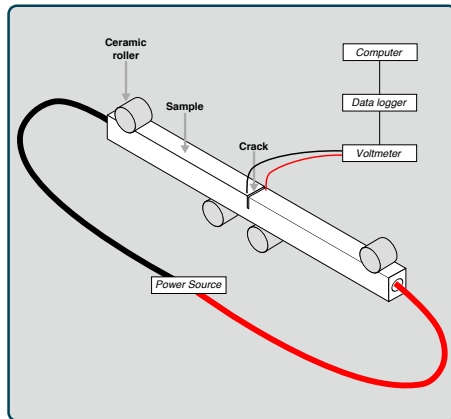
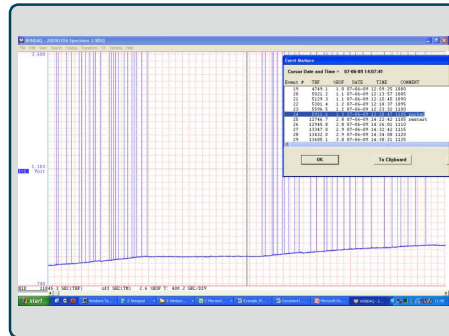
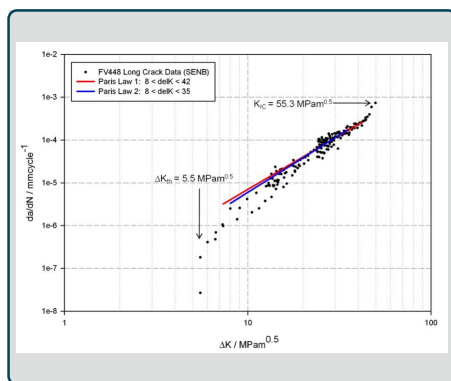


Figure 8: Collecting data during fatigue test

Figure 9:  $da/dN$  versus  $\Delta K$  plot

## Summary

In this case study, the researcher collects potential different readings from a voltmeter whilst applying loads to a sample of material containing a crack. Using this data, along with calibration data, it is possible to know how quickly a crack grows in a material at different levels of applied stress.

This case study provided good examples of a wide range of research data including:

- Collecting data through scientific experiment (the fatigue test);
- Using spreadsheet software to manipulate and refine CSV data by smoothing it;
- How transforming the data is useful for analysing raw data (in this case by generating a plot of stress against number of cycles).

Figure 6.17: The *Introducing Research Data* guide (page 12).

### 3 Aerodynamics: Simulations Using Computational Fluid Dynamics

This section gives an example of data usage in aerospace engineering. This gives a good illustration of using models or simulations to acquire data.

#### Data categories in this case study

- Case study provides good example
- Also relevant in case study

##### Sources of data

Models or simulations	●	Air flow simulation in Fluent
Derived data	○	Data produced from simulation
Reference data	○	Properties of air in simulation

##### Forms of research

Electronic text documents	●	Text document describing the simulation
Digital objects	●	Images produced from TecPlot; animations of air flow
Models, algorithms, scripts	●	Meshes; geometries; MATLAB scripts
Software configuration	●	Fluent case files
Software pre-process files	●	Meshes; geometries
Software post-process files	●	Fluent output

##### Electronic representation of data

Textual	●	Text document describing the simulation
Multimedia	○	Images produced from TecPlot; animations of air flow
Software code	○	MATLAB scripts; Fluent functions (UDFs) in C
Software specific	●	Meshes; geometries

#### Data life cycle steps in this case study

##### Data life cycle stages

Collect	●	(Collection of data is through the simulation in the following stages)
Pre-Process	●	Creation of meshes and geometries
Process	●	Fluent simulation
Post-Process	●	TecPlot analysis
Analyse	●	MATLAB turbulence and vortex analysis
Publish	●	The findings of the analysis, e.g. how to improve rotor design
Curate		

Figure 6.18: The *Introducing Research Data* guide (page 13).

### Background

In aerodynamics, computational models are used to understand air flow, for example, around an aeroplane wing or through an air conditioning unit. Simulation of the air flow is known as *Computational Fluid Dynamics* (or *CFD*) and might be used for improving the performance of the wing or efficiency of the air conditioner. This case study details the process of a *rotor wake simulation* which involves modelling the flow of air around a helicopter rotor operating close to the ground.

### Obtaining the data

Software known as *ANSYS Fluent* takes a number of data items in order to simulate the flow of a fluid (in this case the air affected by a helicopter's rotor blade). This includes a *mesh* which describes the volume of air being simulated broken up into a 3D grid of fluid volumes and a *geometry* to describe the shape of the objects in the simulation. In this case study, a method known as the *actuator line method* allows the geometry data to be substituted for formulae that calculate how the rotor blade interacts with the air by producing lift.

The mesh and some *user-defined functions (UDF)* written in C that handle the actuator line method calculations are loaded into Fluent. A Fluent *case file* is created describing the model's and software's configuration and then computation can begin. Usually this occurs on a *high performance computing* cluster of machines. Using 16 CPU cores, simulating 1 second of air flow takes about 60 hours, broken down into time steps of 0.0005 seconds. A data file is saved every 5 time steps representing 0.0025 seconds of simulation. The total data generated is around 300 GB per 1 second of simulated flow.

### Using the data

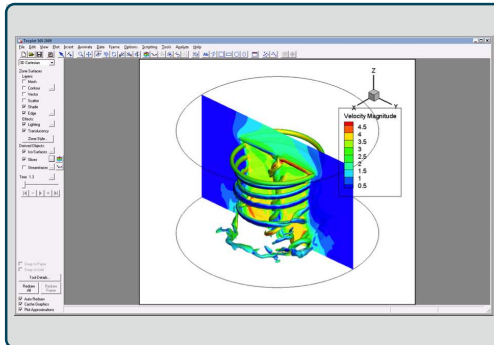
Once the simulation is complete, the data can be used for a number of purposes. Fluent can be used to produce images of the data showing flow development and the mesh quality to assess whether the simulation needs to be rerun with a different configuration or mesh. Software known as *TecPlot* can be used to produce pictures of the processed data either to help identify if the flow is developed and whether to continue the simulation, or to identify the points of interest in the simulation for validation. This qualitative analysis helps to improve the simulation. A program known as *MATLAB* is used to explore the data in a quantitative way for more demanding numerical operations such as turbulence and vortex analysis.

### Looking after the data

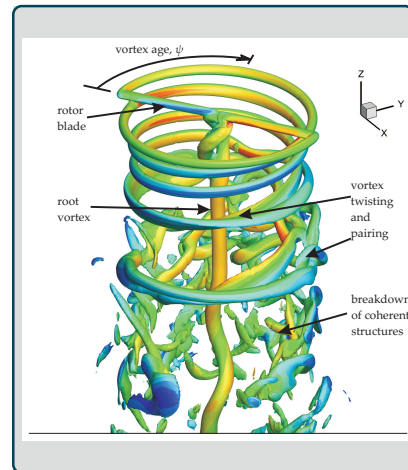
Aerodynamicists have a clear workflow they follow, producing data at each stage. When investigating an extended time period of simulated air flow the researcher may produce a lot of data, which needs looking after. Depending on how long the simulation takes to generate the data, the approach taken by the researcher is different. A simulation taking many months to run produces data that needs more care than one that takes a few days, but in both cases, it is the code that is used to produce the simulation that is the most critical.

Figure 6.19: The *Introducing Research Data* guide (page 14).

**Figure 10:** The Helical Air Flow Produced By A Helicopter Rotor In TecPlot



**Figure 11:** The Air Flow After Analysis As It Might Appear in Published Work



### Summary

The researcher in this case study produces data by creating a mesh and some user-defined functions, and then using software to simulate air flow. The results are then analysed by producing images and videos of the air flow to identify points of interest or assess whether the simulation was sufficient. The results can also be explored with other software to find features such as turbulence and vortices.

This case study provided good examples of a wide range of research data including:

- Collecting data using models or simulations (the Fluent job executed on an HPC cluster);
- Digital objects as data (the images and videos representing the air flow);
- Models, algorithms and scripts (the meshes, geometries, UDFs as C code, and MATLAB scripts);
- Software configuration files (case files for Fluent);
- Software pre-process files (meshes and geometries also come under this category);
- Software post-process files (MATLAB files);
- Software specific data (meshes and geometries are specific to Fluent).

**Figure 6.20:** The *Introducing Research Data* guide (page 15).

## 4 Chemistry: Crystal Structures

We will use manipulating X-ray data to analyse crystal structures as an example for chemistry. This provides a good illustration of deriving data. In this case, taking data from a scientific experiment and correcting and refining to extract only the useful data.

### Data categories in this case study

- Case study provides good example
- Also relevant in case study

#### Sources of data

Scientific experiments	○ The X-ray examination of the crystal
Derived data	● The extraction of $h$ , $k$ and $l$ Miller indices

#### Forms of research

Electronic text documents	● Detail regarding properties of sample
Specimens, samples, artefacts, slides	● The crystalline sample
Digital objects	○ Crystal structure images and videos
Software pre-process files	○ Raw X-ray data
Software post-process files	○ .hk1 data
Experimental results	● Raw X-ray data from diffractometer

#### Electronic representation of data

Multimedia	○ Crystal structure images
Structured	● .hk1 structured text data
Software specific	○ Rigaku's CrystalClear data files
Discipline specific	○ .hk1 data, CIF files
Instrument specific	○ Rigaku diffractometer data

### Data life cycle steps in this case study

#### Data life cycle stages

Collect	● The X-ray examination of the crystal
Pre-Process	
Process	
Post-Process	● The extraction of $h$ , $k$ and $l$ Miller indices
Analyse	● Iterative process to find a model that matches the sample
Publish	● Submit to journal new chemical or new form of known chemical
Curate	● Upload to Crystallographic Data Centre or eCrystals web site

Figure 6.21: The *Introducing Research Data* guide (page 16).

## Background

### Obtaining the data

In chemistry, X-rays are used to examine crystalline samples in order to determine the properties of a newly-created chemical or to analyse the structure of a known chemical. Along with the X-ray data, the history of how the sample was created is required to assist with the analysis.

An X-ray diffractometer is used to perform the scan, in this case manufactured by *Rigaku*. The diffractometer is driven using software supplied by the manufacturer, for example, Rigaku's *CrystalClear*. The instrument produces raw image data made up of 300–400 segments representing a sphere. Each segment's data is stored in a software-specific file of around 4 MB in size, which collectively gives a data set of approximately 1 GB.

The *CrystalClear* software is used to perform corrections on the raw data and execute algorithms to extract the useful data, producing a single structured text file less than 100 MB per data set. This file contains  $h$ ,  $k$  and  $l$  Miller indices of the crystal planes and data relating to spots in the image (known as  $\sigma$  and  $F^2$ ). It is this file that is used for the remaining investigation on the researcher's own workstation outside the laboratory.

### Using the data

This .hkl file can then be used to determine the sample's properties. This is done by producing a model which is assumed to match the characteristics of the sample from which another .hkl file is produced and compared to the sample's. This process of creating a model, producing a comparison .hkl file from the model and then verifying it against the sample's data may be performed many times until the correct model is identified.

The model is then saved in *CIF* format (Crystallographic Information File) which is a discipline specific file format—only a few kB in size—used to store the model, as well as other related data such as the experiment's metadata and even textual data for use in publications.

### Looking after the data

The raw data produced by the X-ray diffractometer is kept indefinitely and is stored on a server in the Chemistry department, mirrored to a secondary server which is then backed up to tape. This raw data may be used again in the future for producing alternative .hkl files if the original file proves insufficient in some way.

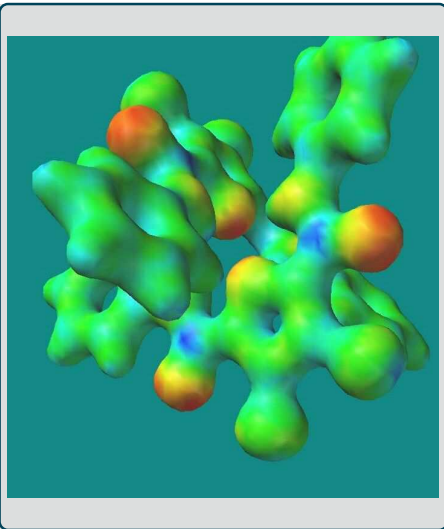
The .hkl, .cif and other files are not stored on the file server as these files are used by the researcher on their own personal computers. It is standard practice to use the experiment ID in the names of these files to ensure they can be traced back to the raw data. Other than that, each researcher is responsible for managing their own working files.

When new chemicals—or a new form of a known chemical—are identified these may be submitted as a paper in a journal. Many journals also now expect the .cif file. The .cif file may also be uploaded to a data centre such as the *Cambridge Crystallographic Data Centre* as a central repository, or shared via a local data repository such as on the *eCrystals* web site.

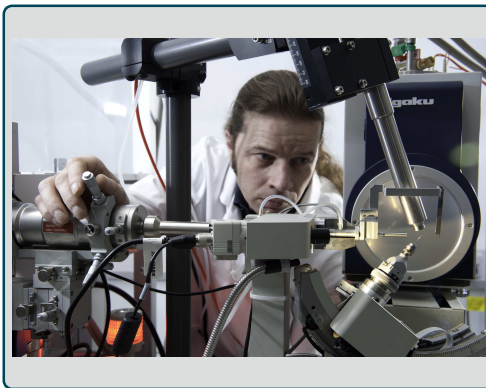
Figure 6.22: The *Introducing Research Data* guide (page 17).



**Figure 12:** A graphical representation of a crystal structure



**Figure 13:** An X-ray diffractometer in use



### Summary

The researcher in this case study produces data relating to a crystalline sample using an X-ray diffractometer. The data is then refined by performing corrections and, using algorithms for analysis, produce the pertinent data needed in the next stage (finding a model that best describes the sample).

This case study provided good examples of a wide range of research data including:

- Collecting data by deriving it from some other form (taking the raw data produced by the X-ray diffractometer and processing it to produce a .hkl file);
- Collecting data from scientific experiment (X-ray diffractometer);
- Using electronic text documents in research (in this case, to describe how the sample was created);
- Specimens and samples (the crystalline specimen being analysed);
- Experiment results (the data produced by the X-ray diffractometer);
- Structured data (.hkl file);
- Software, discipline and instrument specific data.

**Figure 6.23:** The *Introducing Research Data* guide (page 18).

## 5 Archaeology: An Excavation

We will use the data collected by archaeologist during an excavation in this example. This provides a good illustration of observational data as well as reference data.

### Data categories in this case study

- Case study provides good example
- Also relevant in case study

#### Sources of data

Observations	●	Details and features about sites and discoveries
Reference data	○	Maps of an area; records of previous work on a site

#### Forms of research

Electronic text documents	●	Excavation diary
Spreadsheets	●	Spreadsheets detailing finds, <i>e.g.</i> dimensions and weight
Laboratory notebooks, diaries	●	Excavation diary
Audiotapes, videotapes	●	Excavation site video
Photographs, films	●	Photographs of site
Specimens, samples, artefacts, slides	●	Discoveries from site
Digital objects	○	Digital photogrammetry
Database schemas	○	Excavation details database
Database contents	●	Excavation details database
Methodologies, workflows, procedures	●	Excavation procedures
Metadata	●	IPTC photographic metadata

#### Electronic representation of data

Textual	●	Excavation diary
Numerical	●	Spreadsheets detailing finds, <i>e.g.</i> dimensions and weight
Multimedia	●	Photogrammetry; scene visualisations
Structured	●	Excavation database
Software specific	●	<i>ArcGIS</i> files
Discipline specific	●	<i>ARK</i> (Archaeological Recording Kit) files
Instrument specific	●	Polygon Workbench for driving laser scanner

### Data life cycle steps in this case study

#### Data life cycle stages

Collect	●	Taking measurements, photographs and other data created during excavation
Pre-Process		
Process		
Post-Process		
Analyse	●	Assessing collected data to verify nothing was missed; looking for patterns in discovered objects
Publish	●	Publication of discoveries
Curate	●	Uploading to the <i>Archaeology Data Service</i>

Figure 6.24: The *Introducing Research Data* guide (page 19).

### Background

Archaeologists on the Portus Project (University of Southampton, 2012) aim to discover more about the history of Portus, construction of which was begun by Emperor Claudius around 2000 years ago and was the primary port for the city of Rome during imperial times. This case study looks at some of the data collected by an archaeologist on such a project and how it is looked after.

### Obtaining the data

Archaeologists employ many modern techniques such as laser scanning and X-ray computed tomography (CT) as well as traditional approaches such excavation and maintaining a diary.

One of the most important forms of archaeological data comes from observations taken before and during an excavation. This is because the nature of archaeology is destructive and, without good quality records, valuable information could be lost. Not only are the observations important, but the way in which the observations were made are also documented. These records help researchers understand the complete story from discovery to publication and to identify anything missed during a dig.

The accuracy of the observational data affects its usefulness, so researchers try to find ways of being as accurate and detailed as possible, for example by using very precise instrumentation for measuring and scanning. A global positioning system (GPS) or a *total station* containing a theodolite, data logger and distance meter helps with surveying an area, and *digital photogrammetry* is used for recording site layouts in detail.

Descriptions of items of interest, such as the colour and properties of an object, also require precision. For example, to describe colours the *Munsell Book of Color* is used which contains hundreds of reference colours palettes with associated codes. To describe what an object is, archaeologists use a *typology* to categorise it and a thesaurus such as the *Getty Art & Architecture Thesaurus* containing a controlled vocabulary for describing it. Other resources include *INSCRIPTION* provided by the Forum on Information Standards in Heritage containing recommended wordlists for describing sites.

In addition to observational data and the associated documentation, archaeologists use a lot of reference data, especially when planning a dig. This comes from a number of sources including historical records, maps, geophysical data, previous work done at a site and even the researcher's earlier work.

Recording of archaeological data is done using specialist systems such as the *Archaeological Recording Kit* (ARK) which uses a web front-end to populate a database. Another tool is a spreadsheet for recording details of discoveries. Photographs are tagged with the name of the photographer, project, GPS data and other appropriate information using the IPTC metadata standard.

### Using the data

Archaeologists tend to work in teams responsible for different parts of a dig such as surveying, excavating and the team responsible for recording finds. Data collected by one team must be easily accessible and understandable to others to ensure mistakes are not made and important details are not missed.

Figure 6.25: The *Introducing Research Data* guide (page 20).

At the end of a season the excavation is written up. This gives a high level view of the work and helps missing data to be identified as soon as possible. Some researchers make very specific use of the data, for example taking the details of the ceramic objects and analysing their properties, perhaps looking at the relative sizes of all bowls across a site, when they were excavated and where. At a much higher level, some researchers may take data from a number of sites across a country and look for patterns in their distribution.

#### Looking after the data

In order to achieve this level of sharing and thoroughness, archaeologists use specialist software and have strict guidance and standards in place to ensure nothing is missed at any step.

The *Archaeology Data Service* is used as an open-access repository for hosting data. It also provides guidance on how to look after data as well as a place for dissemination of research (University of York, 2012).

Figure 14: The ARK software in use

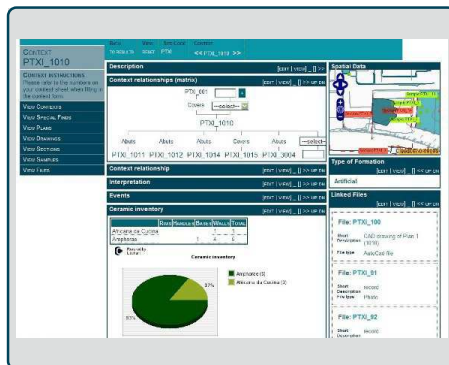


Figure 15: Computer generated simulation of interior of building at Portus



Figure 16: Laser scanning the Cistern complex at Portus using a Leica ScanStation



Figure 17: Taking near infrared photos in Imperial Palace



Figure 6.26: The *Introducing Research Data* guide (page 21).

### Summary

The researcher in this case study produces data relating to an archaeological excavation. The data comes from a number of sources such as a total station, digital photogrammetry, an excavation diary and previous work.

This case study has provided some good examples of the following research data:

- Creating research data by observation (by accurately recording as much detail as possible so data is not lost while excavation takes place)
- Using reference data (by using data from previous digs as well as geophysical and geographical data when planning a dig)
- Using procedures as part of research (archaeologists follow a set of guidelines on how to perform an excavation. If they didn't, mistakes might happen and data might be lost in the process of the dig)
- Specimens, samples and artefacts (found during the excavation)
- Using a database as part of research (to record excavation details)
- Using metadata (metadata is attached to photographs to improve their usefulness)
- Software, discipline and instrument specific data

Figure 6.27: The *Introducing Research Data* guide (page 22).

## Part III

## Data Management Practices

## 1 Storing data

With all the data you will collect as part of your research you need to find a way of looking after that data. Possible ways of doing this include:

- Tagging and search (can you remember the tag later?)
- File naming conventions. Try to use names that match your environment and contain:
  1. Something meaningful to you (such as what you are doing with the file)
  2. Something meaningful to someone else (such as an experiment number or project name)
- Using a database or spreadsheet to track data
- Structuring folders on the disk (one big-flat folder with hundreds of folders versus hierarchical tree)
- Storing it online using a discipline specific repository or archival system (local versus open access)
- Storing it using general repository (*e.g.* EPrints)

## Examples of file naming

**Good**

materialN18\_sample9\_smootheddata.csv

**Bad**

smoothed.csv

## Data formats

Your research data may be used by someone else in the future, possibly even you if you need to refer back to it. It will even become mandatory for research data to be kept for ten years after the last time it was accessed. To prepare for this eventuality, the format data is stored in needs to be considered. After all, the software may not be available any more when the data is next needed.

If you can, try to follow these guidelines:

- Data stored in ASCII text files is likely to still be understandable even if the soft-

Figure 6.28: The *Introducing Research Data* guide (page 23).

ware that generated the file's structure format is no longer available. This includes structured text such as CSV and XML as well as free-form text. Microsoft Word and other word processing software can even export to HTML which can allow some formatting to be retained and provide some protection if the software is no longer available.

- When storing data in other formats, using formats with openly published specifications provides some protection. Even if the software is no longer available, the specifications might permit critical data to be retrieved. For example, the OpenDocument (produced by OpenOffice and LibreOffice as well as others, *e.g.* .ODT) or Office Open XML (produced by Microsoft Office, *e.g.* .DOCX) formats are good choices for storing a word processed document or spreadsheet.
- With figures, using vector formats such as SVG and EPS are often a better choice than raster formats such as JPEG and PNG due to loss of quality incurred when scaling raster images. The SVG specifications is an open standard which may mean it is better suited to long-term archival.
- For documentation, apart from the OpenDocument file formats mentioned previously, PDF could be considered, especially PDF/A which is a standardised version of Portable Document Format that is better suited for long-term archival.

#### File format recommendations

- Textual: OpenDocument Text (.odt), ASCII text (.txt), PDF/A (.pdf), Office Open XML (.docx), and even HTML
- Numerical: Structured text (.csv or .xml), OpenDocument Spreadsheet (.ods), Office Open XML (.xlsx)
- Multimedia: vector (.svg), raster (.tiff or .png), movie (consider using open-source codecs such as Xvid or x264 and non-proprietary containers such as Matroska and Ogg. Note that AVI or MPEG are arguably more supported and are good choice, but licencing and patents may limit their ultimate usefulness.)

## 2 Backups

On 30 October 2005, the University of Southampton's *Mountbatten* building caught fire and many students and staff were suddenly faced with the possibility of having lost months' worth of research. However, data loss more often happens in much less dramatic circumstances. Most of the time, it's a hard drive failing, equipment such as a laptop being lost stolen or simply down to user error, *e.g.* files being deleted, or vendor error, *e.g.* a software bug causing corruption. In order to protect your data, you should ensure

Figure 6.29: The *Introducing Research Data* guide (page 24).

you make regular backups, preferably to more than one physical location.

#### Scans or Photocopies

Research data is not just in the form of electronic data; log books, slides and documents can also be lost. Do not forget these types of research data and ensure you have some form of copy if they are critical to your research.

#### External hard drive

Copying your files onto an external hard drive or even onto another computer is a simple way to backup your data. Note that this does not protect from extreme disaster such as a fire if the hard drive is kept in the same place as the computer. It is a manual process, unless software is configured to do it automatically.

#### Windows Users—SyncToy

For those running Windows machines, *SyncToy* is free software available from Microsoft which will help you set up automatic backups of files across drives.

#### Mac Users—Time Machine

For those running Mac OS X 10.5 or above, purchase an external hard drive and activate *Time Machine* for an easy, automatic solution. For those who run virtual machines, it is a good idea to exclude these files from backups as they will quickly fill your hard drive and backing up a file when open may cause corruption. If you require a Time Machine backup of your VMs, please ensure you use snapshots which Time Machine can backup safely and then you will always be able to roll back to the snapshot.

#### University file store

In the case of a catastrophe, your critical files need to be stored somewhere else. iSolutions provide every student with storage on a server, which is backed up so it is recommended you make use of this. If you have a laptop, you may want to store files locally, rather than on a server so you will have to look into more complicated solutions such as *rsync* or *SyncToy* to help you to keep the files in both places up to date.

#### Non-free alternatives

For more cautious people, a service such as CrashPlan or BackBlaze is useful for backups. CrashPlan and BackBlaze charge a few pounds a month and provide backup software and unlimited backup space on their servers.

DropBox is for synchronisation of files between computers and sharing of data and also has paid plans and a small amount of space is given for free. It should not be relied on for backups as it is not easy to restore all files to a specific point in time to recover from corruption which is something you may require from a backup solution. It is useful for keeping files synchronised between machines and could help minimise data loss in the case of theft.

Figure 6.30: The *Introducing Research Data* guide (page 25).



### 3 Version management

When dealing with large amounts of files, and changes are being made, it is important to follow a process that allows you to cope with all the different versions of the files. For example, you may want to try making a change to a CFD mesh to investigate the effect it has on the simulation. These changes may not have the desired effect so it may be necessary to “roll-back” to an earlier version. There are a few techniques to doing this and it is important to pick one that works for you and allows you to manage your research data in a way you are comfortable with but still protects you.

- Backup your files before editing to permit you to make a simple roll-back if necessary
- Store multiple copies of your files using some naming convention to permit easy identification of the different versions

When dealing with large amounts of files which have frequent and complex changes, it may be worth investing the time learning a more advanced technique:

- Use software that handles versioning of your files for you. Good examples are *Mercurial*, *Subversion* and *Git*. There are many views on which is better. *Git* and *Mercurial* use similar approaches and are probably the most lightweight and the easiest to get going with. *Subversion* has a steeper learning curve but it is common and the University hosts a *Subversion* server which may be preferable to installing it yourself.

Figure 6.31: The *Introducing Research Data* guide (page 26).

## Part IV

## Acknowledgements

- The categorisation of research data collection was defined in Research Information Network (2008).
- The forms of research data and categorisation of electronic storage of research data was adapted from The University of Edinburgh (2011).
- The main authors of this document were Mark Scott, Richard Boardman, Philippa Reed and Simon Cox in the Faculty of Engineering and the Environment.
- The following people helped with the preparation of this document:
  - Andy Collins (Human Genetics case study)
  - Thomas Mbuya and Kath Soady (Materials Fatigue Test case study)
  - Gregory Jasion (CFD case study)
  - Simon Coles (Chemistry case study)
  - Graeme Earl (Archaeology case study)
- We acknowledge ongoing support from the University of Southampton, Robert's funding, The DataPool project, Microsoft, EPSRC, BBSRC, JISC, AHRC and MRC.

## References

- Broad Institute (2012), 'Integrative Genomics Viewer'. [Accessed 23-Mar-2012].  
URL: <http://www.broadinstitute.org/software/igv/>
- Digital Curation Centre (2010), 'DCC Curation Lifecycle Model'. [Accessed 20-Feb-2012].  
URL: <http://www.dcc.ac.uk/resources/curation-lifecycle-model>
- Humphrey, C. (2006), 'e-Science and the Life Cycle of Research'. [Accessed 20-Feb-2012].  
URL: <http://datalib.library.ualberta.ca/humphrey/lifecycle-science060308.doc>
- Research Information Network (2008), 'Stewardship of digital research data: a framework of principles and guidelines'. [Accessed 20-Feb-2012].  
URL: <http://www.rin.ac.uk/our-work/data-management-and-curation/stewardship-digital-research-data-principles-and-guidelines>
- The University of Edinburgh (2011), 'Defining research data'. [Accessed 20-Feb-2012].  
URL: <http://www.ed.ac.uk/schools-departments/information-services/services/research-support/data-library/research-data-mgmt/data-mgmt/research-data-definition>
- University of Southampton (2012), 'Portus Project'. [Accessed 15-Mar-2012].  
URL: <http://www.portusproject.org/>
- University of York (2012), 'Archaeology Data Service'. [Accessed 13-Mar-2012].  
URL: <http://archaeologydataservice.ac.uk/>

Figure 6.32: The *Introducing Research Data* guide (page 27).

**Table 6.1:** Feedback from research data management introduction lectures.

Question	Lecture 1:	Lecture 2:
	80 students Month 7	30 students Month 2
Would students change what they were doing as a result of the lecture?	60/80	15/30
Should the talk be given to future postgraduates?	78/80	20/30
Did any students feel it was a complete waste of time?	2/80	1/30

## 6.4 Feedback received

The material has been presented a number of times across the university. This section discusses some of the feedback received during these classes.

### 6.4.1 Faculty of Engineering Sciences and the Environment

The talk has been given twice to students at the Faculty of Engineering Sciences and the Environment at the University of Southampton. The audience was entirely made up of postgraduates, with audience numbers given in Table 6.1. To gauge its reception, students were asked three questions at the end of the lecture, as shown in Table 6.1. Most found the guide was useful with only a few students feeling it was not relevant to their research. The majority believed it should be given to first year postgraduates in future. Interestingly, the feedback was more positive when the lecture was given in month seven, once postgraduates had settled in. This is perhaps because they had begun their research and had started to collect data.

The guide itself proved very popular and an additional print run was required due to unexpected demand.

### 6.4.2 WebScience Doctoral Training Centre

The audience of approximately fifteen was predominately Computer Science postgraduates, but there were some from the humanities such as archaeology. In this case, the computer scientists felt that they were already aware of the best practices and pitfalls, and felt the material was of limited value. Having said that, some felt that the issues needed discussing so the material could be adapted for more advanced audiences. The humanities disciplines felt it was relevant to them.

### 6.4.3 Gradbook training event

The audience contained eight students across a number of disciplines. The feedback was mainly positive from this talk. Part III had the best reception (best practices) and the students from the computer science discipline in this talk also felt that being made to think about data preservation was valuable, even where they were already aware of some of the issues. Evaluation forms returned by the students (six out of eight) confirmed that the talk should

be given again with 100% answering ‘Yes’ or ‘Maybe’ to the question ‘Would you recommend this event to a colleague/fellow student?’ (Byatt et al. 2013).

## 6.5 Evaluation

This was our first attempt at this type of guide. It was felt that future versions should extend the third section (‘Data Management Practices’) to provide more tips on data management, as students found this part the most useful as reference material and reported that it helped them to think about how to look after their own data. The case studies in Part II (‘Case Studies’) provided perspective and encouraged students to think about data in general. Part I (‘Five ways to think about research data’) is useful to set the scene, but should be kept brief in a lecture to maintain the students’ interest. The case studies enable researchers from diverse disciplines to engage and relate and the feedback received from students suggests that thinking about these issues partway through Year 1 of their research study is helpful and necessary.

Given the feedback, lectures require tailoring to the audience and it seems sensible to not mix students who have different levels of understanding of the issues. Some of the concepts discussed may seem elementary to, for example, computer science postgraduates who have probably studied them earlier in their careers. The order of the sections in the guide seems appropriate, but the order in which the lectures are given could be possibly changed to ensure that the highest impact part of the material is presented at the beginning whilst the interest of the students is at its highest (i.e. Part III). Since publication of the guide, the lecture has started giving more advice about how data might be accessed again in twenty years, such as about exporting some files to HTML which is ASCII but allows formatting to be preserved and discussing the pros and cons of file metadata (tagging) in greater detail. Further detail that might be appropriate includes the recommendations discussed in Section 2.3.1 that were not part of the guide: using standardised file formats, regular data refreshing (copying of data to new storage media) to protect against data degradation, transformation/migration of data from older formats into newer formats, emulation of older software to access obsolete data formats, and more information about providing metadata about the data to describe its meaning to future generations.

Producing a guide on research data for students and training them on the issues of data management is one approach. It may also be useful to convert the material into a web-based guide for online training although this has already been attempted elsewhere, discussed in the next section.

## 6.6 Summary

This chapter has presented the educational material used to introduce research data management to students at the University of Southampton. Research data management has become a requirement for many researchers and ensuring that students start considering these problems early in their careers should help them as the amount of data used in research continues to rise.

While Chapter 3 discussed data management tools for research data at the end of its life cycle, and Chapters 4 and 5 showed tools to manage research data whilst still in use, this chapter has looked at data across its entire life cycle and shown one approach to training researchers to ensure the value of data is understood by its creators.

Teaching about the management of research data is now part of the University's institutional data management strategy and 10-year road map with the guide forming part of the online training resources (University of Southampton Library 2014).

## Chapter 7

# Summary and outlook

---



HIS THESIS PRESENTED a flexible system that has been used for the management of materials engineering and medical data and metadata, and extended to link data to a publication. Educational material used for introducing research data management has also been presented. This chapter discusses improvements that could be made to the model, the research questions asked in Chapter 1 and whether the objectives have been met.

### 7.1 Discussion of objectives

Section 1.7.1 presented the objectives for this thesis. In this section, we will take each one in turn and discuss whether the objective was met.

**To develop a platform-independent framework for the management of heterogeneous data and metadata:** The model of file system, file system watcher, database and data management interface presented in Chapters 4 and 5 was shown to cope with heterogeneous data and metadata with a number of case studies from materials engineering and medicine disciplines. The model's simple schema, shown in Figure 5.1, uses non-proprietary data types which would permit the use of other database management systems. File system events can be generated by other operating systems, such as FSEvents in Mac OS X (Singh 2006, 1416) and inotify in Linux (Love 2005), which would enable a file system watcher to be created for other platforms. There is much flexibility in the choice of platform for interacting with the metadata database as shown by the testing script in Section 4.4.1, the SPARQL end-point in Section 5.2.8 and the querying examples in Python and C# in Section 4.6.5.

**To implement this framework with off-the-shelf technology, to build an extensible system that naturally fits into the existing workflows of users:** The implementation of this framework used a combination of the NTFS file system on a Microsoft Windows file server, Microsoft SQL Server, Microsoft SharePoint and the .NET Framework.

Apart from the costs for the Microsoft Windows server, the architecture of the framework allowed the use of free off-the-shelf technology for the implementation, achieving the objective whilst reducing the level of investment required to install the system: the development environment used free editions of SQL Server and SharePoint (Microsoft SQL Server 2008 (SP3) Express Edition and Microsoft SharePoint Foundation 2010); the NTFS 3.1 file

**Table 7.1:** Current number of data sets in the production version of the Heterogeneous Data Centre.

Data Set Type	Data Sets	Files	Parameters
Creep test	34	34	138
Fatigue test	22	25	160
Material data sheet	32	24	578
Template	3	0	0
No type	14	920	11
<b>Totals</b>	<b>105</b>	<b>1003</b>	<b>887</b>

system has been supported by Microsoft Windows computers since 2001 (Microsoft 2001, 2007); and .NET Framework version 4 used by the file system monitor has been available as a free download for Windows computers since 2010 (Microsoft 2010b).

It was possible to manage the types of data discussed in this thesis, even with editions of some products with reduced capabilities, by storing the data files in the file system separately from the metadata: whilst the Express edition of Microsoft SQL Server has a maximum database size restriction of 10 GiB (Microsoft 2014a), the NTFS file system is able to cope with file sizes of nearly 16 TiB since Microsoft Windows 2008 (Microsoft 2014b, 2003) permitting the 10 GiB of space for the database to be retained for purely metadata.

The additional features gained through the use of the paid for versions of these products could be utilised if required: the deployed version of the metadata database in Section 5.3 used a non-free version of SQL Server (Microsoft SQL Server 2008 R2 (SP2) Standard Edition) provided by the institution's central IT service to benefit from their managed backups. This also provided additional features for better performance and reliability: more memory, larger databases, failover clustering and access to database tuning management tools (Microsoft 2014a).

The use of a file system for storing data files has become well known to users, ever since the early network file systems that were discussed in Section 2.4.3 became available, and is how the objective of naturally fitting into the existing workflows of users was achieved.

The extensibility of the framework was demonstrated in Section 5.3 with a number of plug-ins.

### **To develop the framework's implementation with real data and feedback from users from materials engineering:**

The Heterogeneous Data Centre was deployed at the University of Southampton and a number of users from materials engineering were involved in its testing, as discussed in Chapters 4 and 5. These users have since added data to a production installation of the system, and the current number of data sets hosted in the system are listed in Table 7.1. In addition, a University of Southampton student project used the HDC to compare creep within nickel alloys (Austin 2014). This demonstrates that the objective to develop the system with the use of real data has been achieved.

Feedback from these users has driven its development throughout, from the requirements collection in Section 2.5 to user feedback discussed in Section 4.7.1 and implemented in Chapter 5. Appendix C provides evidence of some user feedback.

**To explore tools that allow users to query, report on and work with deposited data:**

Chapters 4 and 5 describe tools and reports that users can use to query and work with their data. Sections 5.4.2–5.4.4 evaluated the features with respect to ‘metadata and metadata tools’, ‘data set discoverability’ and ‘data set tools adding value to users’. Features that allowed this objective to be met included manual and automatic metadata import functionality, data file viewing and parsing tools, report generation and SQL, Python, C# and SPARQL querying of data.

**To test the framework’s implementation with other types of data:** Section 5.3 showed the framework in use with data collected from the human genetics discipline, with plug-ins built specifically for this data.

**To integrate the system with established document publishing technologies to demonstrate data management throughout its lifecycle:** The EP2DC project showed how this was possible in Chapter 3. Once the system was more mature, the EP2DC web service layer was rewritten to work with the HDC project and tested in Section 5.3.3.

**To develop educational material to encourage users to begin considering the issues of data management:** Chapter 6 presents the educational material that was used in the Faculty of Engineering at the University of Southampton. Section 6.4 shows that, although improvements to this early attempt are possible, education of users on the significance of their data is necessary.

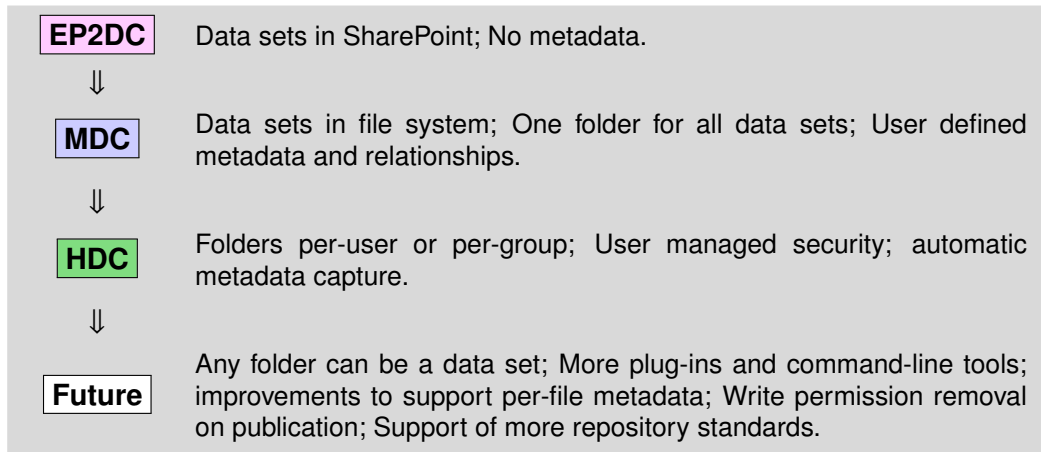
## 7.2 Discussion of research questions

A number of research questions were asked in Section 1.7.1. This section takes each question in turn and discusses how the question was addressed.

**Without disrupting users’ existing workflows, what is required to capture the prepublication data and metadata of materials engineers whilst it is still in use?:** Chapter 4 presented a data model which has been used as a data staging repository for managing materials engineering data, and has shown its use with a number of use cases. The exploitation of a file system for data storage allowed the support of a wide range of types of data including microfocus computed tomography data sets where the files can be in the order of gigabytes. The metadata database with its nesting abilities and templating permitted storage of complex metadata and production of reports. This was supported by the publication Scott, Boardman, Reed, Austin et al. (2014).

**Given the heterogeneity of materials engineering data, can this framework cope with future needs and be adapted for use in different disciplines?:** The model was extended in Chapter 5 with additional plug-ins, more advanced reports, better security features necessary for managing heterogeneous data and automatic metadata ingestion, and was tested with





**Figure 7.1:** Data centre evolution, between EP2DC, MDC and HDC projects and future possibilities.

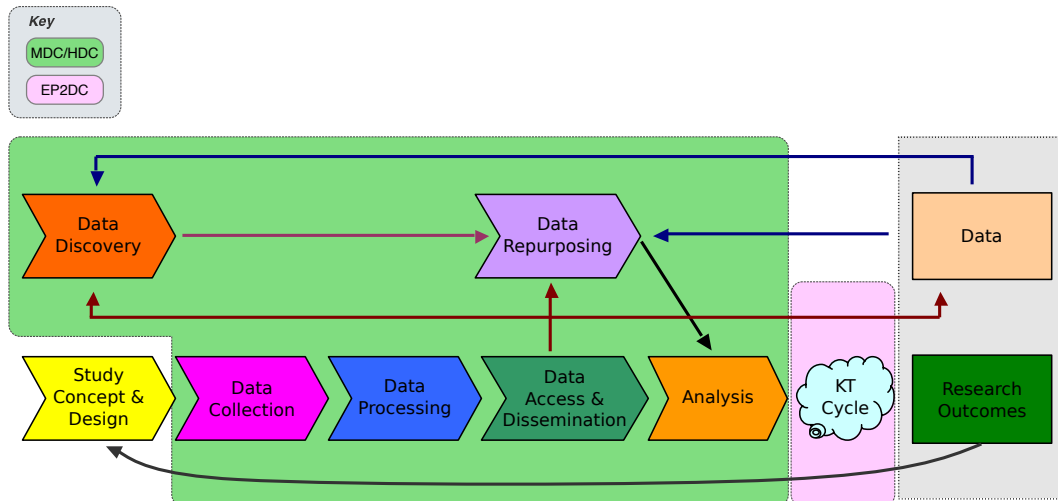
a limited amount of medical data. We also looked at features aimed at enticing users to deposit data into the file server's file system which enable data and metadata capture early in its life cycle. Enhancements for future versions were discussed, particularly more flexibility on what the system recognises as a data set and improvements to support per-file metadata. This was supported by the publication Scott, Boardman, Reed and Cox (2014).

#### **How would data in such a system be linked to a research publication at a later date?:**

With the EP2DC project in Chapter 3, we also considered data at the end of its life cycle showing how data could be transmitted to a remote data repository by a document repository when uploading a publication and how that data can be validated before acceptance. This was demonstrated with an early version of the data repository with documents stored in Microsoft SharePoint, and then again with the more mature data repository in Chapter 5 with documents stored in a file system. Figure 7.1 illustrates the evolution of the data centre between projects, and how each improved on the previous iteration. This was supported by the publication Austin et al. (2011).

#### **Is it necessary to educate users about data management issues and how would this be done?:**

Chapter 6 showed teaching material as a way of encouraging researchers to start thinking about the management of their data. We reported on the feedback from the lectures that have been completed and concluded that users do find this material helpful to start them thinking about how to protect their data in the long term. We also discussed improvements that could be made to this material. Once users realise the importance of their data and metadata, it is hoped that they will be attracted to data staging repositories such as the HDC which will allow data to be captured early in its life cycle so it is ready for transfer to a data archive repository – along with captured metadata – or be linked to a publication using a tool such as EP2DC at the end of its life cycle. This was supported by the publication Scott, Boardman, Reed and Cox (2013b) and the conference Scott, Boardman, Reed, Byatt et al. (2013).



**Figure 7.2:** The data life cycle as illustrated by Humphrey (2006). The stages have been highlighted to indicate which areas have been addressed by the HDC and EP2DC projects; the entire life cycle was discussed in Chapter 6.

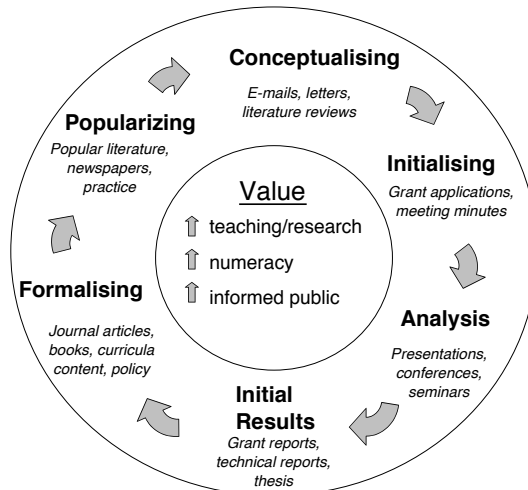
## 7.3 Outlook

The majority of research concentrated on building reports and interface components in the Microsoft SharePoint system, which was the priority for this project. The API examples in Section 4.6.5 suggest that there is more power and flexibility in driving the system from a command line program or cluster job submission script than we have explored fully here.

More testing with other types of data (perhaps using the subject areas discussed in the guide in Chapter 6) would be desirable, which would inevitably lead to more features and increased reliability, and adding direct support for protocols such as SWORD, OAI-ORE, OAI-PMH and OData could be considered sensible future work to improve the interoperability of the system. The educational material contained case studies of data usage of five researchers from medicine, archaeology, aerodynamics, materials engineering and chemistry. This is biased towards scientific subjects and, in future, case studies for other subjects should be considered to cover a wider audience.

The use of a file system for data storage provides some interesting possibilities as the system can benefit from improvements in file system technologies including through the use of replicated or distributed file systems. The synchronisation routines of the file system monitor could be expanded to allow the metadata in the database to be duplicated in the file system providing several additional benefits including: a backup of the metadata; the ability for tools to be built that can operate on the metadata even when the database is unavailable or the data set has been removed from the system; and the ability to replicate data and metadata to other data centres simply by copying the data set. The monitor already places a small amount of data in each data set for tracking purposes and to allow changes made while offline to be reconciled; we could extend this to include the metadata.

Figure 7.2 shows one common illustration of the data life cycle (Humphrey 2006) which was used in the *Introducing Research Data* guide in Section 6. The HDC model is aimed at



**Figure 7.3:** The knowledge transfer cycle ('KT Cycle') in Figure 7.2 (Humphrey 2006).

the earlier stages highlighted in the figure, whereas the EP2DC starts to address the *KT Cycle* stage in Figure 7.2 and expanded in Figure 7.3 where research outcomes are communicated.

To link these two worlds, our data model would need to be adapted in order to prevent data being changed once published: integration with the file system watcher could be used to remove a user's write permissions to their data set in the same way as users share data sets (see Section 5.2.3). DOI minting would be required to track the published data set and metadata exchange between repositories would also be desirable. To tackle data provenance, it is relatively straightforward for the file system monitor to trigger backups of the data sets in the file system any time a data file is modified, ensuring historical versions of data are retained. Other solutions include file system snapshots or even the use of revision control systems with automatic commits to track changes. Plug-ins for the HDC interface could then allow users to restore from these backups or even produce reports that compare data from different points in time.

It is proposed that managing in-use data and metadata is just as important to users as published data. Appropriate education of users and a data staging repository with a flexible and extensible data model supports this without precluding the ability to publish the data at a later date.

## Appendix A

# User requirements report questionnaire results

---



SET OF QUESTIONS were produced to collect user requirements. This appendix summarises the response to the questions. The work in this section is taken from Rimer and Reed (2009).

**1. Does your research focus on specific materials? Yes/No**

74% of users answered yes.

**Which ones and with what objectives?**

Most common alloys named here were Aluminium (20%), Titanium (20%), Nickel (15%), Steel (20%) and CFRP (25%). Less specific answers were given in the website responses.

**2. Do you already use standardised testing techniques? Yes/No**

77% of users answered yes.

**If so which ones and would you want to specify new testing methodologies?**

BSI and ASTM were the most common sources for standardisation. These predominantly included fatigue, tensile, hardness, surface roughness, three point bending, four point bending, long and short crack tests and Charpy impact. Users also reported use of ISOs.

**3. Does your research depend on working with experimental data? Yes/No**

80% of users answered yes.

**Measured (observational, experimental) or derived and for what purpose e.g. FEA, modelling, data mining?**

The most common answer by far here, with 54% of users citing it as a purpose, was FEA. Most users were interested in measured data, but there were also a proportion (18%) interested in derived data.

**4. Do you ever use data from other sources? Yes/No**

74% of users answered yes.

**What is your experience of the availability, quality, and usefulness of this sort of data?**

Essentially the outcome of this question showed that the availability, quality and usefulness of this data were highly variable. It is fair to say that for the most part that simple data (common values) was much easier to find than more difficult data (see appendix of comments), such as high quality images.

5. **Is there ever a requirement to augment your data from other sources, such as literature and web-enabled databases? Yes/No**

71% of users answered yes.

**What are the circumstances?**

The most common answers here were that data augmentation is used either in the literature review, or as a comparison for results achieved by the user.

6. **Would you find it useful to have a universal method of storing data? Yes/No**

75% of users answered yes.

**What type of data would you find it useful to store and to whom would you like to make it available?**

It is fair to say users were unsure of the possibilities and viability here. To get a clear list of deliverables a much larger sample size is required, as there is currently little commonality of requirements established.

7. **Have you ever needed to use experimental data from earlier years and/or previous projects? Yes/No**

76% of users answered yes.

**What are your experiences locating and using such data?**

Overwhelming response (57%) reporting finding this data was difficult and time consuming.

8. **Have you ever shared your data either on request or in the context of collaboration? Yes/No**

67% of users answered yes.

**What have been your experiences and did/would you have any reservations about sharing your data?**

The major concerns here were confidentiality and acknowledgement, although there were also concerns over the commercial value of data. Mutually beneficial relationship required.

9. **Do you have security concerns for the data you would be adding to the data centre? Yes/No**

46% of users answered yes.

**What are the major issues?**

Out of all the questions this was the question with the lowest percentage of users answering yes. Commercial sensitivity was a major concern in this area, however sharing published data was not considered an issue.

10. **Do you consider conservation of experimental data worthwhile? Yes/No**

87% of users answered yes.

**What do you think would be the advantages and disadvantages?**

The question had the most positive response of all the questions. Here users wanted the facility in order to back-up their work, but also believed it would prevent the repetition

---

of work and therefore create greater continuity. Concerns were related to the cost and space requirements.

11. **Would you be prepared to trial the data centre? Yes/No**

66% of users answered yes.

**If you supplied data to the centre how do you envisage it being used and what constraints would you need to apply?**

Most users agreed there would need to be some sort of restriction on data access, however to get a clearer view of the uses envisaged again a greater sample size is required.

12. **In general would you like to see the engineering materials sector moving towards a more consistent/standardised data layout? Yes/No**

83% of users answered yes.

**If this is the case what would you ideally like to find in a data centre and what are your major concerns as a user?**

Once again for this question there were lots of concerns over feasibility, with the majority of users agreeing it's a good idea in an ideal world. There were also concerns that standardisation would inhibit progression.



## Appendix B

# MatDB schema user interviews



MatDB IS A STANDARD SCHEMA able to represent data from tensile, creep and fatigue tests as XML (Ojala and Over 2008). As discussed in Section 2.5.3, a set of interviews were held to discuss the MatDB Schema using the figures in the following pages. A jigsaw piece was used to show where the users must refer to another diagram. Using these jigsaw pieces, the user was able to build up a picture of the entire schema. An XML schema is defined with elements, types and attributes. Types define the data used in the schema and can be simple, such as an integer, or complex, such as a combination of other elements and attributes.

The MatDB schema starts with the top-level element in Figure B.1 which uses the *Materials*, *Source* and *Tests* types represented by jigsaw pieces to indicate they are defined elsewhere. The rest of the schema is made up of *Symbol*, *Specimen*, *Temperature*, *TestCondition*, *ReferenceToReport*, *Fracture*, *Stress* and *OrientationDeviationAngle* elements which are defined as top-level elements and then referenced by the main types.

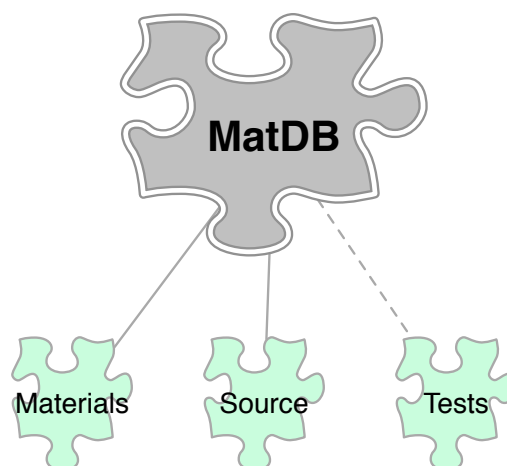


Figure B.1: *MatDB* top-level element.



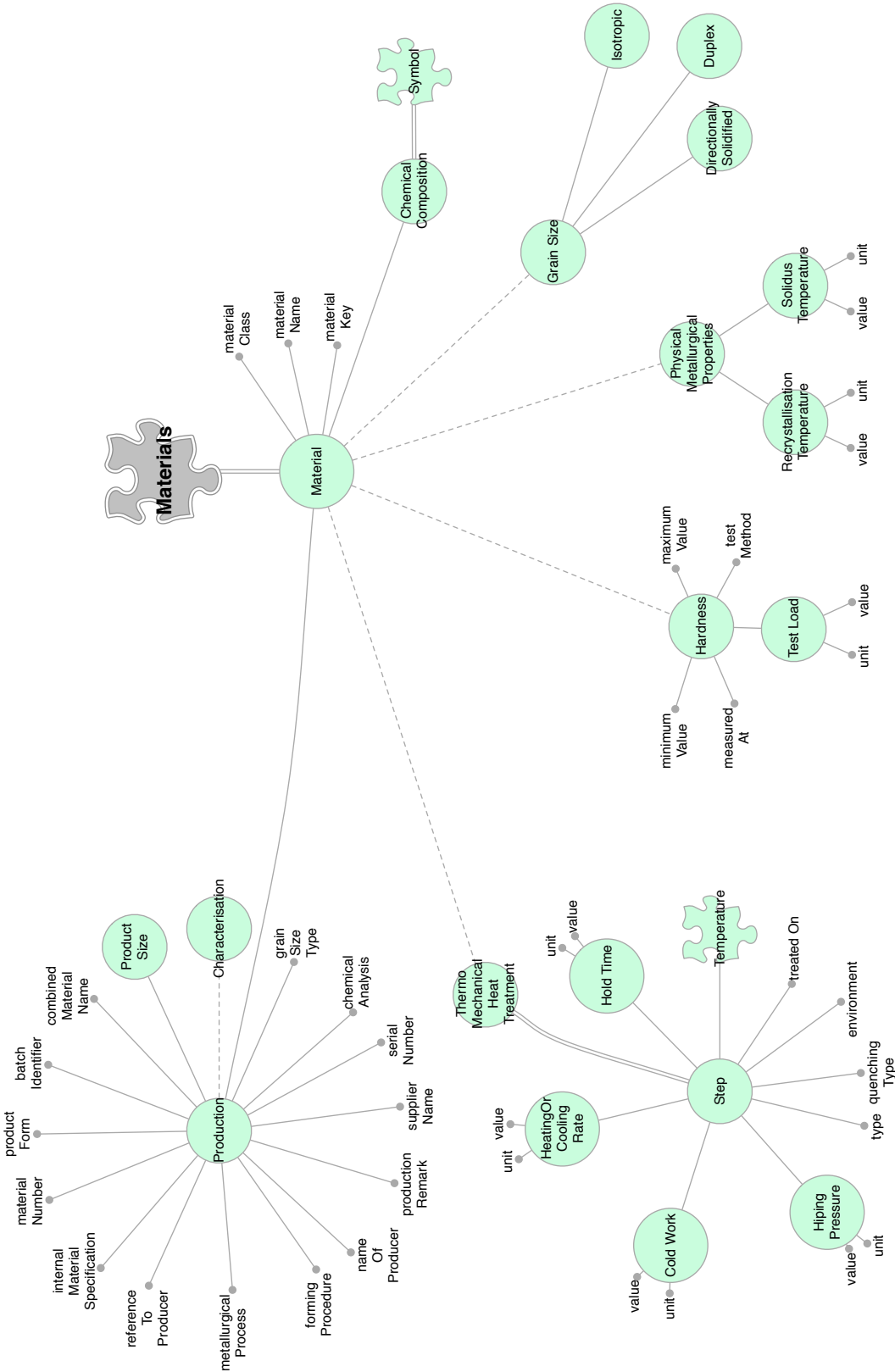


Figure B.2: *Materials* type.

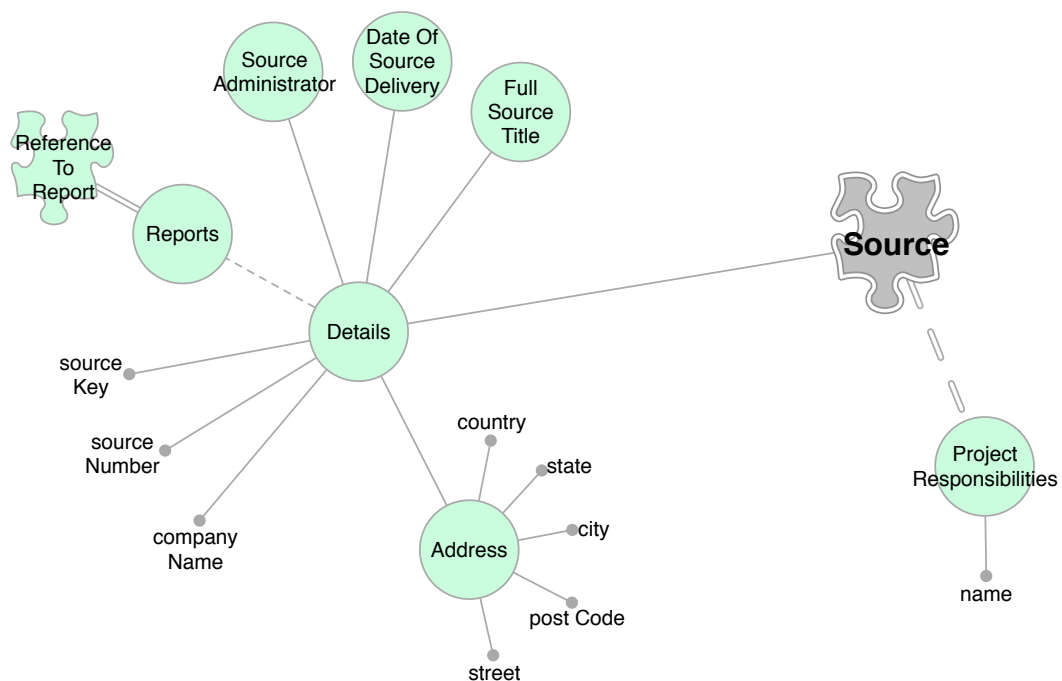


Figure B.3: Source type.

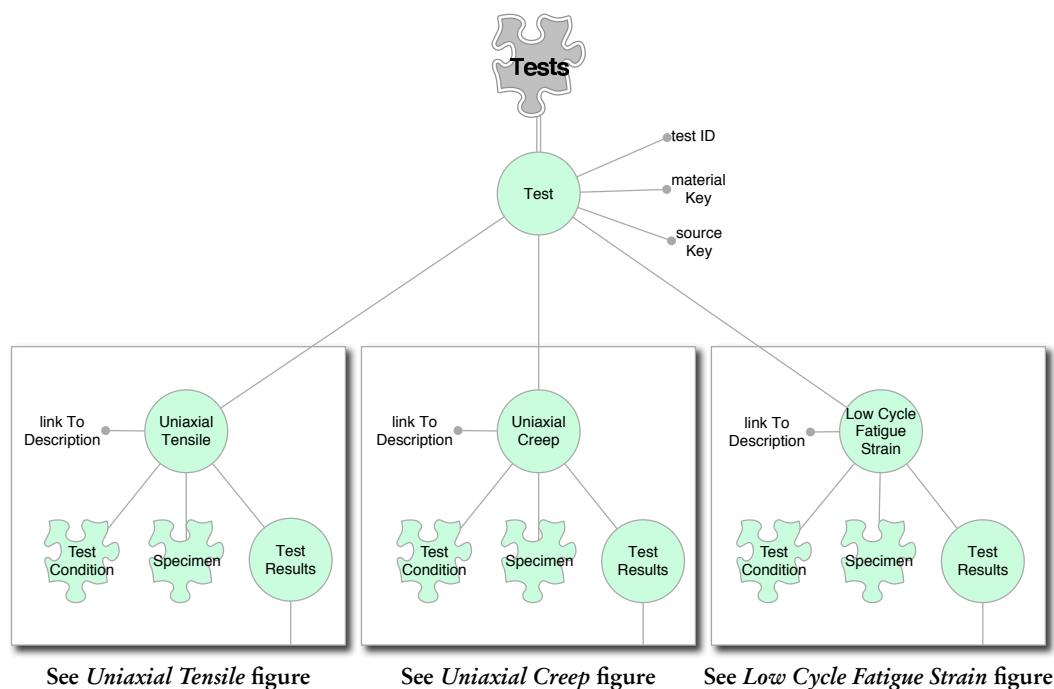


Figure B.4: Tests type. This type is large so it has been divided into four diagrams to help with comprehension. This figure shows how the Tests type fits together. For *Uniaxial Tensile*, see Figure B.6. For *Uniaxial Creep*, see Figure B.5. For *Low Cycle Fatigue Strain*, see Figure B.7.

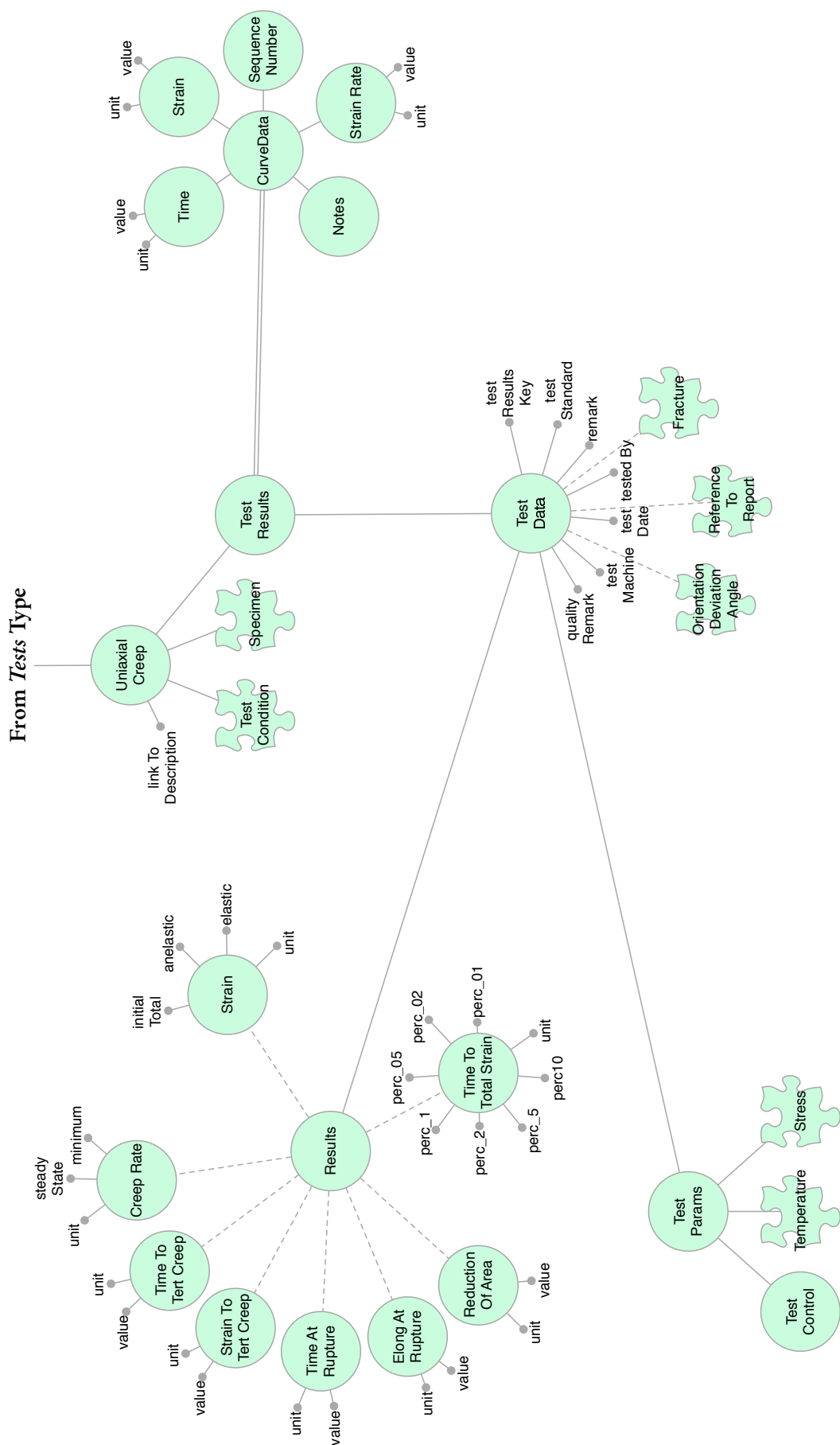


Figure B.5: Uniaxial Creep test data (Tests type continued).

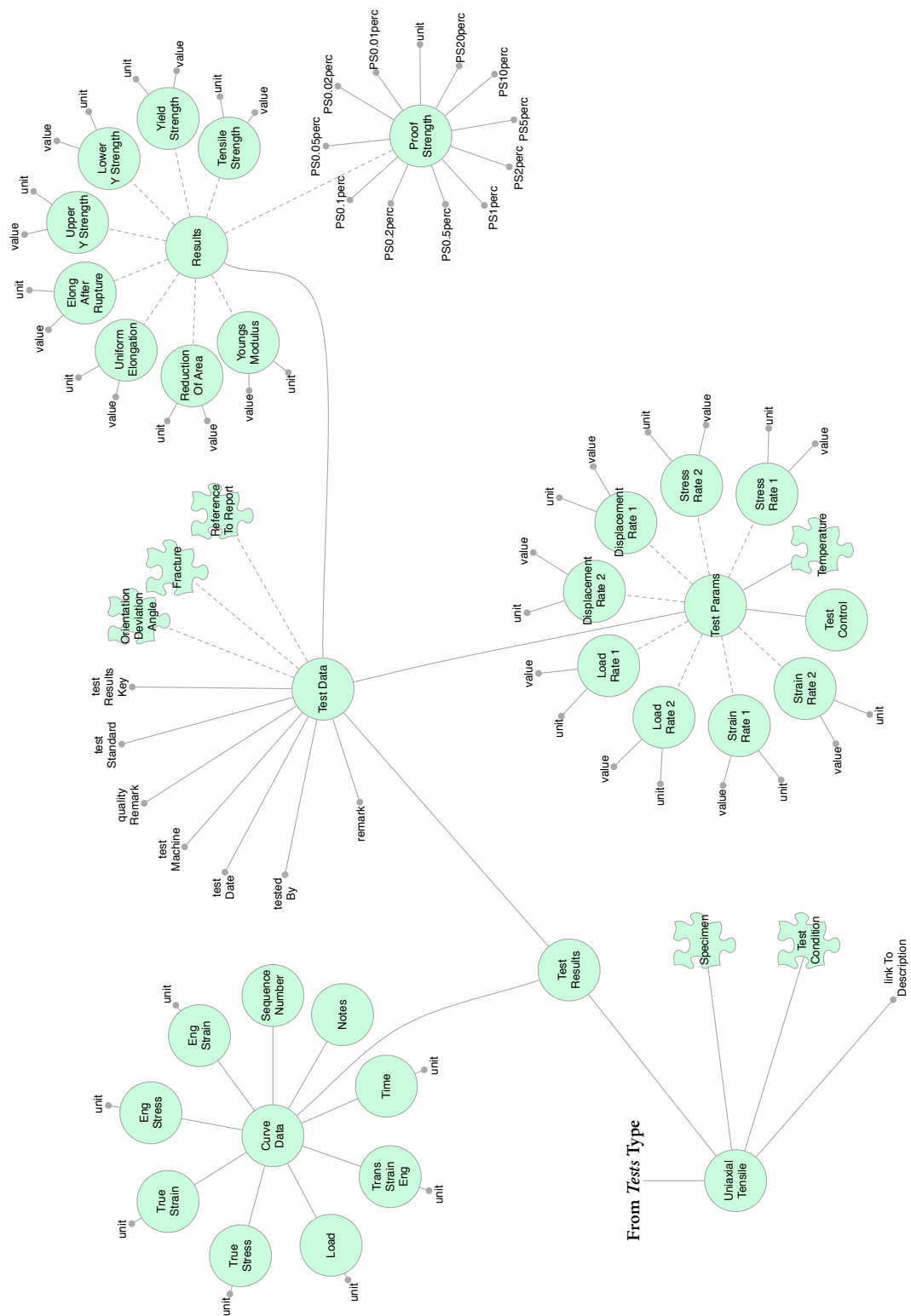


Figure B.6: Uniaxial Tensile test data (*Tests* type continued).

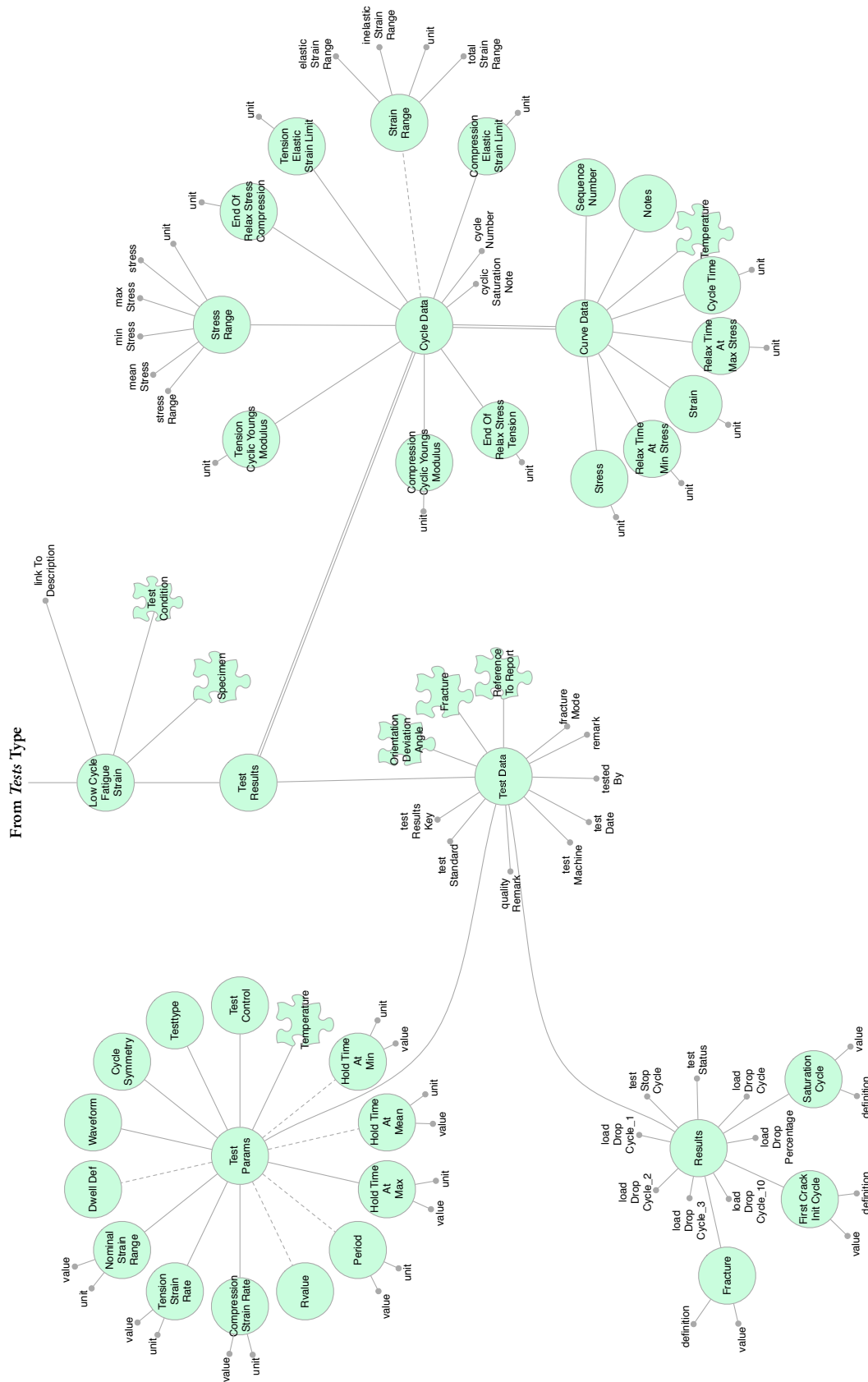


Figure B.7: *Low Cycle Fatigue Strain* test data (*Tests* type continued).

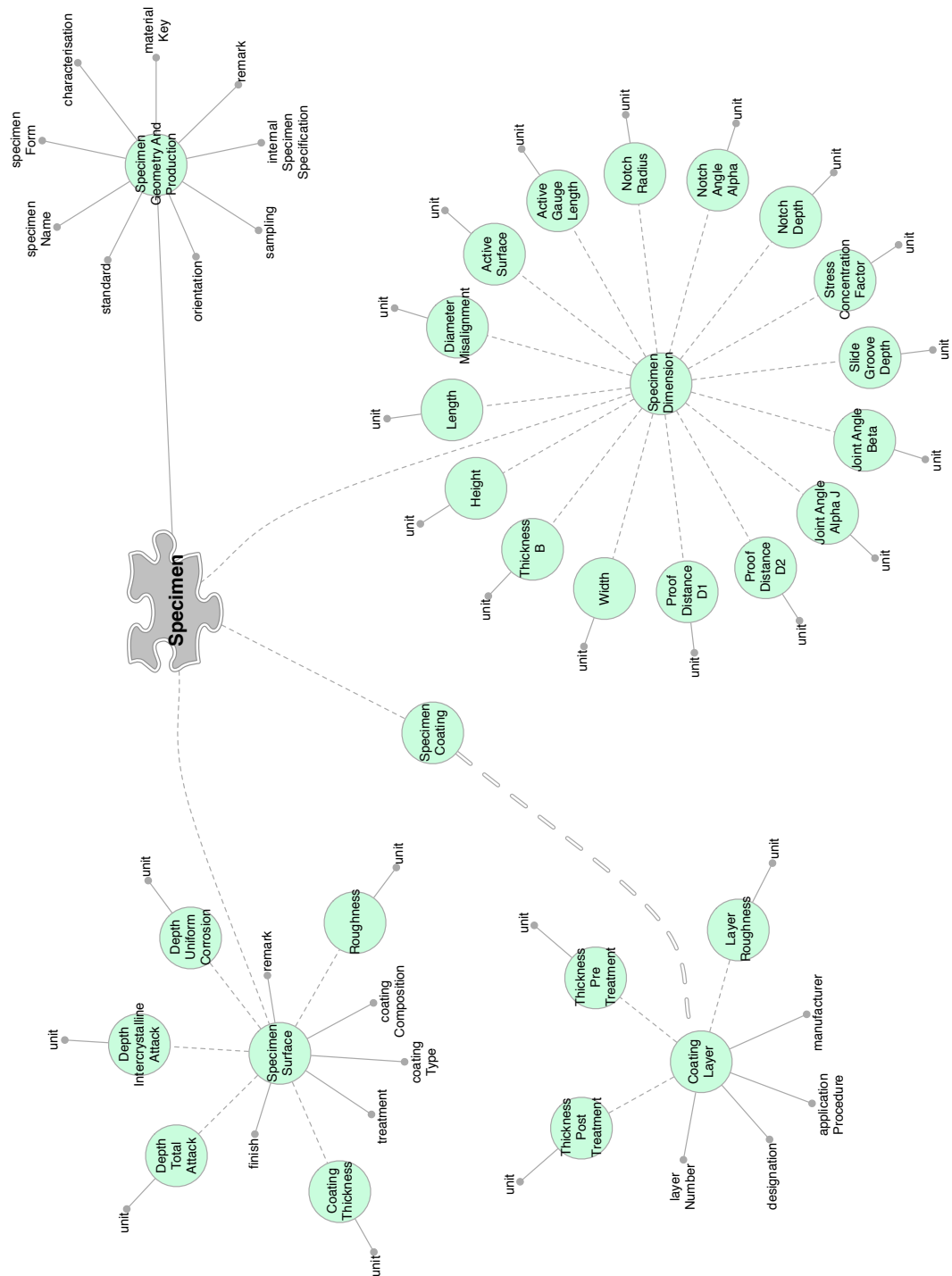


Figure B.8: *Specimen* element.

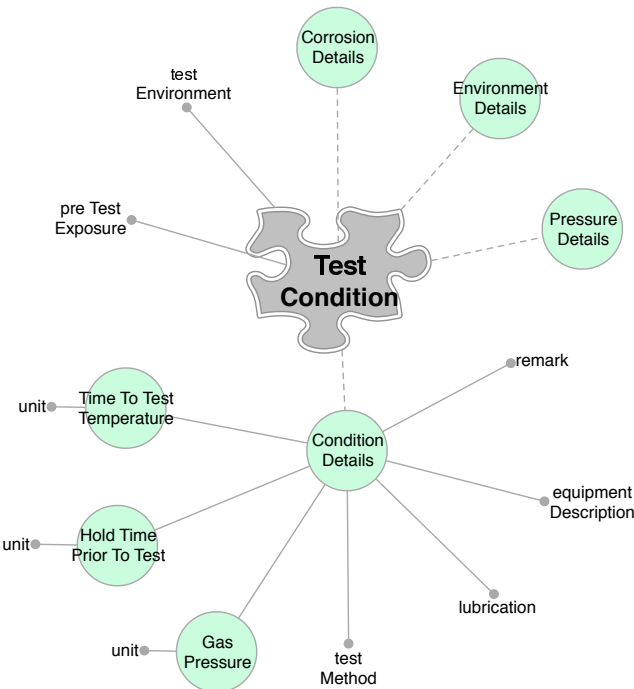


Figure B.9: TestCondition element.

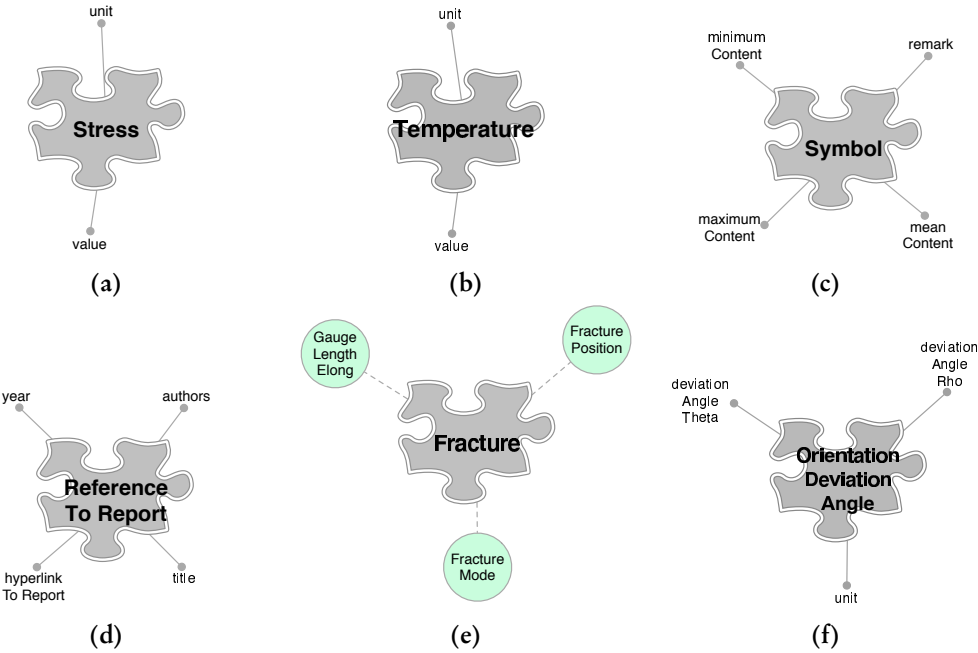


Figure B.10: Other elements: (a) Stress, (b) Temperature, (c) Symbol, (d) Reference To Report, (e) Fracture, (f) Orientation Deviation Angle.

## Appendix C

# User feedback

---



HIS APPENDIX PROVIDES personal communications that show some of the feedback from users involved in the system's development. They have been reproduced with permission.

### C.1 Email containing user feedback

From: Soady K.A.  
Sent: 27 February 2013 18:10  
To: Scott C.M.  
Subject: RE: MDC login details

Hi Mark,

You will see that I have now created two experiments and linked them using a collection, I have a couple of questions and some suggestions which I wondered if perhaps we could discuss if you are available next Friday afternoon (from 3pm), if not , perhaps you could answer them via return email:

1. I have structured my experiment files such that I also have my raw data in them in a sub-folder. This seemed sensible to me at the time, but for some of my experiments I have literally thousands of csv files (I haven't uploaded that one yet). The problem is that I can see all the csv files in the experiment, which makes the experiment very big; I wondered if there was a way of hiding files you didn't necessarily need / want to see all the time? This also appears to slow the experiment down a lot in terms of loading and refreshing and typing in descriptions for the files etc. Or is the only way round this to upload the raw data in a separate experiment and then use the collections function?
2. I would like to upload a table to the metadata area so it prints as part of the report. I would also like to reorder the metadata in terms of the sequence it appears in the report now I have entered it, but can't seem to work out how to do either of these. I'm sure we discussed tables last time we met - can you remind me how to add one please? **[Reordering functionality added]**
3. I would like to re-order the figures so they appear in a logical sequence in the report rather than in the order of the filenames. It would also be useful to be able to caption the figures with the filename as well as the description since adding a description is rather slow once you have lots of files in an experiment! **[Filename added to caption]**



4. I renamed an experiment, but the old name still appears, now crossed out when I log in. I can see a recycle bin, but I can't work out how to put the experiment in it! I know you say to delete by deleting from the HDC folder, but I'm not sure how to get rid of this other experiment now I have renamed it in the HCD folder. **[File system monitor algorithms improved]**

Once I have figured out how to get round some of these issues I will have another go at trying to organise my data. . .

Many Thanks

Kath

## C.2 Email containing summary of meeting with a user and their feedback

From: Scott C.M.  
Sent: 22 November 2013 16:42  
To: austin t.  
Subject: IP Ball Rolling

Hi Tom

I hope the meeting/tutorial today was useful. Here are the minor issues we found today, which I have now fixed:

- 1 - A float in an integer parameter not being highlighted as a possible **[Fixed]**
- 2 - Copying from a template other than the first one on the drop down menu always copying from first one on the list **[Fixed]**
- 3 - Not being able to download certain files with unusual extensions **[Security settings now relaxed]**
- 4 - Parameters occasionally not saving. This turned out to be because we had clicked 'Show all children' AND edited a parameter value. The 'Show all children' checkbox rebuilt the parameter grid, overriding the fact that we had made an edit. **[Fixed]**

If you get a chance, hopefully you can test these fixes and let me know if you come across any other oddities.

**[Personal information removed...]**

Mark

## C.3 Email containing summary of meeting with a user and their feedback

From: Scott C.M.  
Sent: 20 February 2014 13:56  
To: austin t.

Subject: HDC additions

Hi Tom

Here are a list of additions we discussed at our meeting today:

1 - Although links are one way, it would be useful to find a way of making this easier for users. This could be done as two separate list of links in and links out, or perhaps to simplify the interface removing that distinction from the user. **[Directional links added]**

2 - Parameter entry across data sets would be invaluable when depositing multiple data sets. Entry via Excel seems like a good solution - saving to CSV - with the data set ID as the column heading and the parameter name as the row heading. Sub parameters would be grouped under a [SubParameter] with only one level of nesting supported in this way. Automatic generation of the CSV for editing could make creation of this file easier.

3 - XY scatter graph parameter selection would be easier if it was more like the parameter selection in the Category parameter graph. **[Improved method of selecting parameters]**

4 - Add Parameter IDs to parameter list to make it easier to know the ID of a parameter **[Fixed]**

**[Personal information removed...]**

Mark

## C.4 Email containing user feedback and summary of experience

From: austin t.

Sent: 28 February 2014 12:45

To: Scott C.M.

Subject: HDC Feedback

Hi Mark

**[Personal information removed...]**

Pros:

- The HDC has enabled me to effectively store a range of complex materials data in a uniform format from a range of sources.
- The set-up was exceptionally quick and provided a solid basis from which data entry could occur.
- Data input was intuitive and adjustments could be easily made from a good user interface.
- Direct comparisons could be made for parameters from different data sets more easily than using other software.

Cons:

- A repetitive cycle was required for data entry - There was no 'one table to many files' function that would have sped up the process.

- The comparison between multiple parameters from data was limited. Whilst achievable by manual methods, it would be an improvement to be able to select variables and plot in a 2-D scatter and 3-D format. **[Improved parameter selection for 2D scatter plot]**

Overall, the system has enabled me to complete a task, that has not been done before, in a simple and easy-to-follow manner. The current capability from the HDC provides a good ground for further detailed analysis of large engineering topics with scope to have more functions built in to give the user a fast, reliable and efficient way to collect and compare data from multiple sources.

**[Personal information removed...]**

Tom

# Bibliography

---

- Adobe Systems Incorporated. 2005. *XMP specification*. September. (Cited on page 18).
- . 2008. *Document management – Portable Document Format – Part 1: PDF 1.7*. 1st ed. Edited by Technical Committee ISO/TC 171. Adobe Systems Incorporated, 1st July. (Cited on page 18).
- Åkerman, Johan. 2005. ‘Toward a universal memory’. *Science* 308, no. 5721 (22nd April): 508–10. doi:10.1126/science.1110549. (Cited on page 20).
- Ali, M. S., P. A. S. Reed and S. Syngellakis. 2009. ‘Comparison of fatigue performance of HVOF spray coated and conventional roll bonded aluminium bearing alloys’. *Materials Science and Technology* 25, no. 5 (1st May): 575–81. doi:10.1179/174328408X322213. (Cited on page 25).
- Allan, Chris, Jean-Marie Burel, Josh Moore, Colin Blackburn, Melissa Linkert, Scott Loynton, Donald MacDonald et al. 2012. ‘OMERO: flexible, model-driven data management for experimental biology’. *Nature Methods* 9, no. 3 (28th February): 245–53. doi:10.1038/nmeth.1896. (Cited on page 48).
- Allinson, Julie, Les Carr, Jim Downing, David F. Flanders, Sebastien François, Richard Jones, Stuart Lewis, Martin Morrey, Glen Robson and Neil Taylor. 2008. *SWORD AtomPub profile version 1.3*. Accessed 28th April 2014. <http://purl.org/net/sword/>. (Cited on page 44).
- Allinson, Julie, Sebastien François and Stuart Lewis. 2008. ‘SWORD: Simple Web-service Offering Repository Deposit’. *Ariadne* (54). issn: 1361-3200, accessed 29th April 2014. <http://www.ariadne.ac.uk/issue54/allinson-et-al/>. (Cited on page 44).
- Altini, V., F. Carena, W. Carena, S. Chapeland, V. Chibante Barroso, F. Costa, R. Divià et al. 2010. ‘The ALICE electronic logbook’. *Journal of Physics: Conference Series* 219, no. 2 (April): 022027. issn: 1742-6596. doi:10.1088/1742-6596/219/2/022027. (Cited on page 50).
- Amazon. 2010. *Amazon S3 website*. Accessed 4th June 2010. <http://aws.amazon.com/s3/>. (Cited on page 22).
- . 2013. *Amazon SimpleDB documentation*. Accessed 31st October 2013. <http://aws.amazon.com/documentation/simplydb/>. (Cited on page 51).
- American National Standards Institute. 1986. *Coded character sets – 7-bit American Standard Code for Information Interchange*. ANSI X3.4, March. (Cited on page 13).

- Anderson, J. Chris, Jan Lehnardt and Noah Slater. 2010. *CouchDB: The definitive guide*. O'Reilly. ISBN: 978-0-596-15589-6. (Cited on page 41).
- Anderson, Thomas E., Michael D. Dahlin, Jeanna M. Neefe, David A. Patterson, Drew S. Roselli and Randolph Y. Wang. 1996. 'Serverless network file systems'. *ACM Transactions on Computer Systems* 14, no. 1 (February): 41–79. doi:10.1145/225535.225537. (Cited on page 22).
- Antelman, Kristen, Emily Lynema and Andrew K. Pace. 2006. 'Toward a twenty-first century catalog'. *Information Technology and Libraries* 25, no. 3 (September): 128–39. ISSN: 2163-5226. doi:10.6017/ital.v25i3.3342. (Cited on page 45).
- Aritome, S., R. Shirota, G. Hemink, T. Endoh and F. Masuoka. 1993. 'Reliability issues of flash memory cells'. *Proceedings of the IEEE* 81, no. 5 (May): 776–88. ISSN: 0018-9219. doi:10.1109/5.220908. (Cited on page 20).
- Armbrust, Michael, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee et al. 2010. 'A view of cloud computing'. *Communications of the ACM* 53, no. 4 (April): 50–58. ISSN: 0001-0782. doi:10.1145/1721654.1721672. (Cited on pages 22, 23).
- Aroush, D. Raz-Ben, E. Maire, C. Gauthier, S. Youssef, P. Cloetens and H. D. Wagner. 2006. 'A study of fracture of unidirectional composites using in situ high-resolution synchrotron X-ray microtomography'. *Composites Science and Technology* 66, no. 10 (August): 1348–53. ISSN: 0266-3538. doi:10.1016/j.compscitech.2005.09.010. (Cited on page 25).
- ASTM Standard E647. 2013. *Standard test method for measurement of fatigue crack growth rates*. West Conshohocken, PA. doi:10.1520/E0647-13. <http://www.astm.org>. (Cited on page 27).
- Atlassian. 2013. *CONF-20912: Large attachments (more than 2 GB) cannot be uploaded*. Confluence bug report. Accessed 9th June 2013. <https://jira.atlassian.com/browse/CONF-20912>. (Cited on page 49).
- Austin, Tim, Mark Scott, Steven Johnston, Philippa Reed and Kenji Takeda. 2011. 'EP2DC – an EPrints module for linking publications and data'. In *Proceedings of PV 2011: Ensuring Long-Term Preservation and Adding Value to Scientific and Technical Data*. Toulouse, France, 15th November. (Cited on pages xix, 9, 11, 53, 154).
- Austin, Tom. 2014. *Curation and data mining approaches to creep within nickel based super-alloys*. Master of Engineering Individual Project Final Report. University of Southampton, 27th March. (Cited on page 152).
- Austin, Tom, and Mark Scott. 2014. User feedback meeting, 20th February. Faculty of Engineering and the Environment, University of Southampton, United Kingdom. (Cited on page 75).

- AV&IT Standardization Committee. 2010. *JEITA CP-3451C: Exchangeable image file format for digital still cameras: Exif Version 2.3*. Japan Electronics and Information Technology Industries Association, April. (Cited on page 18).
- Barnard, G. A., III, and L. Fein. 1958. 'Organization and retrieval of records generated in a large-scale engineering project'. In *Papers and Discussions Presented at the Eastern Joint Computer Conference: AIEE-ACM-IRE '58 (Eastern)*, 59–63. Philadelphia, Pennsylvania: ACM, 3rd–5th December. doi:10.1145/1458043.1458058. (Cited on page 21).
- Baru, Chaitanya, Reagan Moore, Arcot Rajasekar and Michael Wan. 1998. 'The SDSC Storage Resource Broker'. In *Proceedings of the 1998 conference of the Centre for Advanced Studies on Collaborative Research: CASCON '98*, 5. Toronto, Ontario, Canada: IBM Press, 30th November–3rd December. (Cited on page 22).
- Bass, Michael J., and Margret Branschofsky. 2001. 'DSpace at MIT: Meeting the challenges'. *Joint Conference on Digital Libraries*: 468. doi:10.1109/JCDL.2001.10093. (Cited on page 42).
- Beagrie, Neil. 2000. 'The JISC digital preservation focus and the digital preservation coalition'. *New Review of Academic Librarianship* 6 (1): 257–67. doi:10.1080/13614530009516815. (Cited on page 4).
- Beall, Jeffrey. 2005. 'Metadata and data quality problems in the digital library'. *Journal of Digital Information* 6 (3). ISSN: 1368-7506. (Cited on page 18).
- Begeman, K., A. N. Belikov, D. R. Boxhoorn, F. Dijkstra, H. Holties, Z. Meyer-Zhao, G. A. Renting, E. A. Valentijn and W.-J. Vriend. 2011. 'LOFAR Information System'. *Future Generation Computer Systems* 27 (3): 319–28. ISSN: 0167-739X. doi:10.1016/j.future.2010.08.010. (Cited on page 47).
- Benson, Dennis A., Ilene Karsch-Mizrachi, David J. Lipman, James Ostell and Eric W. Sayers. 2010. 'GenBank'. *Nucleic acids research* 38, no. suppl 1 (January): D46–51. ISSN: 1362-4962. doi:10.1093/nar/gkp1024. (Cited on page 49).
- Bent, Graham, Patrick Dantressangle, David Vyvyan, Abbe Mowshowitz and Valia Mitsou. 2008. 'A dynamic distributed federated database'. In *Proceedings of the 2nd Annual Conference of International Technology Alliance: ACITA '08*. (Cited on page 40).
- Berners-Lee, Tim. 2006. *Linked Data*, 27th July. Accessed 10th August 2014. <http://www.w3.org/DesignIssues/LinkedData.html>. (Cited on page 46).
- Bilderback, Donald H., Pascal Elleaume and Edgar Weckert. 2005. 'Review of third and next generation synchrotron light sources'. *Journal of Physics B: Atomic, Molecular and Optical Physics* 38, no. 9 (14th May): S773–S797. doi:10.1088/0953-4075/38/9/022. (Cited on page 29).

- Boardman, Richard P. 2005. ‘Computer simulation studies of magnetic nanostructures’. PhD diss., School of Engineering Sciences, Faculty of Engineering, Science and Mathematics. (Cited on page 20).
- Bonwick, Jeff, Matt Ahrens, Val Henson, Mark Maybee and Mark Shellenbaum. 2003. ‘The Zettabyte File System’. In *Proceedings of the 2nd USENIX Conference on File and Storage Technologies: FAST ’03*. San Francisco, CA: USENIX Association, 31st March–2nd April. (Cited on page 21).
- Brase, J. 2009. ‘DataCite – A global registration agency for research data’. In *Proceedings of the 4th International Conference on Cooperation and Promotion of Information Resources in Science and Technology: COINFO ’09*, 257–61. Beijing: IEEE, 21st–23rd November. doi:10.1109/COINFO.2009.66. (Cited on page 4).
- Bray, Tim, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler and François Yergeau, eds. 2008. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. W3C Recommendation. World Wide Web Consortium, 26th November. Accessed 29th April 2014. <http://www.w3.org/TR/2008/REC-xml-20081126/>. (Cited on page 13).
- British Standards Institution. 2003. *BS ISO 12108:2002. Metallic materials. Fatigue testing. Fatigue crack growth method*. British Standards Institution, 21st November. ISBN: 978-0-580-42962-0. (Cited on page 27).
- Brumfiel, Geoff. 2011. ‘High-energy physics: Down the petabyte highway’. *Nature* 469 (20th January): 282–83. doi:10.1038/469282a. (Cited on page 50).
- BSON specification. 2011. Accessed 19th July 2011. <http://bsonspec.org>. (Cited on page 41).
- Buffiere, J.-Y., E. Maire, C. Verdu, P. Cloetens, M. Pateyron, G. Peix and J. Baruchel. 1997. ‘Damage assessment in an Al/SiC composite during monotonic tensile tests using synchrotron X-ray microtomography’. *Materials Science and Engineering: A* 234–236 (30th August): 633–35. ISSN: 0921-5093. doi:10.1016/S0921-5093(97)00302-X. (Cited on page 28).
- Byatt, Dorothy, Mark Scott, Gareth Beale, Simon J. Cox and Wendy White. 2013. ‘Developing researcher skills in research data management: Training for the future – A DataPool project report’. Project Report. Accessed 26th May 2014. <http://eprints.soton.ac.uk/351026/>. (Cited on pages xx, 149).
- Byron, Angela, Addison Berry, Nathan Haug, Jeff Eaton, James Walker and Jeff Robbins. 2008. *Using Drupal*. O’Reilly Media. ISBN: 978-0-596-51580-4. (Cited on page 42).
- Cabinet Office. 2012. *Open data white paper: Unleashing the potential*. White paper Cm 8353. UK Government, June. (Cited on page 46).
- Callister, W. D., and D. G. Rethwisch. 2009. *Materials science and engineering: An introduction*. John Wiley & Sons. ISBN: 978-0-470-55673-3. (Cited on page 26).

- Cardoso, Jorge. 2007. 'The Semantic Web vision: Where are we?' *Intelligent Systems, IEEE* 22, no. 5 (September): 84–88. ISSN: 1541-1672. doi:10.1109/MIS.2007.4338499. (Cited on page 45).
- Carr, Leslie A., Lee Dirks, Michele Kimpton and Sandy Payette. 2009. 'Strategies for innovation and sustainability: Insights from leaders of open source repository organizations'. In *Proceedings of the 4th International Conference on Open Repositories: OR 2009*. Atlanta, USA, 18th May–21st May. HDL: 1853/28441. (Cited on page 43).
- Chang, Fay, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes and Robert E. Gruber. 2008. 'Bigtable: A distributed storage system for structured data'. *ACM Transactions on Computer Systems* 26, no. 2 (June): 4:1–4:26. ISSN: 0734-2071. doi:10.1145/1365815.1365816. (Cited on page 51).
- Chodorow, Kristina, and Michael Dirolf. 2010. *MongoDB: The definitive guide*. O'Reilly Media. ISBN: 978-1-4493-8156-1. (Cited on page 41).
- Church, George M., Yuan Gao and Sriram Kosuri. 2012. 'Next-generation digital information storage in DNA'. *Science* 337, no. 6102 (28th September). doi:10.1126/science.1226355. (Cited on page 20).
- Cock, Peter J. A., Christopher J. Fields, Naohisa Goto, Michael L. Heuer and Peter M. Rice. 2010. 'The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants'. *Nucleic Acids Research* 38 (6): 1767–71. doi:10.1093/nar/gkp1137. (Cited on page 103).
- Codd, Edgar Frank. 1970. 'A relational model of data for large shared data banks'. *Communications of the ACM* 13, no. 6 (June): 377–87. doi:10.1145/362384.362685. (Cited on page 37).
- Coles, Simon J., Jeremy G. Frey, Michel B. Hursthouse, Mark E. Light, Andrew J. Milsted, Leslie A. Carr, David DeRoure et al. 2006. 'An e-Science environment for service crystallography – from submission to dissemination'. *Journal of Chemical Information and Modeling* 46, no. 3 (24th February): 1006–16. ISSN: 1549-9596. doi:10.1021/ci050362w. (Cited on pages 43, 47).
- Cox, S. J., R. P. Boardman, L. Chen, M. Duta, H. Eres, M. J. Fairman, Z. Jiao et al. 2002. 'Grid services in action: Grid enabled optimisation and design search'. In *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing: HPDC-11*. Edinburgh, Scotland, 24th July–26th July. doi:10.1109/HPDC.2002.1029943. (Cited on page xx).
- Cox, Simon J., James T. Cox, Richard P. Boardman, Steven J. Johnston, Mark Scott and Neil S. O'Brien. 2014. 'Iridis-pi: A low-cost, compact demonstration cluster'. *Cluster Computing* 17, no. 2 (22nd June): 349–58. ISSN: 1386-7857. doi:10.1007/s10586-013-0282-7. (Cited on page xx).



- Crockford, D. 2006. *The application/json Media Type for JavaScript Object Notation (JSON)*. RFC 4627. Network Working Group, July. Accessed 27th March 2014. <http://www.ietf.org/rfc/rfc4627.txt>. (Cited on page 41).
- Curdt, Constanze, Dirk Hoffmeister, Guido Waldhoff and Georg Bareth. 2008. ‘Spatial data infrastructure for soil-vegetation-atmosphere modelling: Set-up of a spatial database for a research project (SFB/TR32)’. In *Proceedings of the XXIst International Society for Photogrammetry and Remote Sensing Congress*, vol. XXXVII Part B4, 131–36. Beijing, China: The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 3rd–11th July. (Cited on page 48).
- Curdt, Constanze, Dirk Hoffmeister, Guido Waldhoff, Christian Jekel and Georg Bareth. 2012a. ‘Development of a metadata management system for an interdisciplinary research project’. In *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, I-4:7–12. Melbourne, Australia: XXII ISPRS Congress, 25th August–1st September. doi:10.5194/isprsannals-I-4-7-2012. (Cited on page 48).
- . 2012b. ‘Scientific research data management for soil-vegetation-atmosphere data: The TR32DB’. *International Journal of Digital Curation* 7 (2): 68–80. doi:10.2218/ijdc.v7i2.208. (Cited on page 48).
- Daconta, Michael C., Leo J. Obrst and Kevin T. Smith. 2003. *The Semantic Web: A guide to the future of XML, web services, and knowledge management*. John Wiley & Sons, 20th June. ISBN: 978-0-471-43257-9. (Cited on page 45).
- Daley, R. C., and P. G. Neumann. 1965. ‘A general-purpose file system for secondary storage’. In *Proceedings of the Fall Joint Computer Conference, Part I: AFIPS ’65 (Fall, part I)*, 213–29. Las Vegas, Nevada: ACM, 30th November–1st December. doi:10.1145/1463891.1463915. (Cited on page 21).
- Date, C. J. 2004. *An introduction to database systems*. Pearson/Addison Wesley. ISBN: 978-0-321-19784-9. (Cited on page 37).
- DeCandia, Giuseppe, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Voshall and Werner Vogels. 2007. ‘Dynamo: Amazon’s highly available key-value store’. *ACM SIGOPS Operating Systems Review* 41, no. 6 (December): 205–20. ISSN: 0163-5980. doi:10.1145/1323293.1294281. (Cited on page 51).
- Deelman, Ewa, Gurmeet Singh, Miron Livny, Bruce Berriman and John Good. 2008. ‘The cost of doing science on the cloud: The Montage example’. In *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing: SC ’08*, 50:1–50:12. Austin, Texas: IEEE Press, 15th–21st November. ISBN: 978-1-4244-2835-9. (Cited on page 22).

- Dietrich, Dianne. 2010. 'Metadata management in a data staging repository'. In 'Metadata for Scientific Data', *Journal of Library Metadata* 10 (2–3): 79–98. doi:10.1080/19386389.2010.506376. (Cited on page 43).
- Dinu, Valentin, and Prakash Nadkarni. 2007. 'Guidelines for the effective use of entity-attribute-value modeling for biomedical databases'. *International Journal of Medical Informatics* 76, nos. 11–12 (November–December): 769–79. ISSN: 1386-5056. doi:10.1016/j.ijmedinf.2006.09.023. (Cited on page 47).
- Divià, R., U. Fuchs, I. Makhlyueva, P. Vande Vyvre, V. Altini, F. Carena, W. Carena et al. 2010. 'The ALICE online data storage system'. *Journal of Physics: Conference Series* 219, no. 5 (April): 052002. ISSN: 1742-6596. doi:10.1088/1742-6596/219/5/052002. (Cited on page 50).
- Dominguez-Sal, D., P. Urbón-Bayes, A. Giménez-Vañó, S. Gómez-Villamor, N. Martínez-Bazán and J. Larriba-Pey. 2010. 'Survey of graph database performance on the HPC scalable graph analysis benchmark'. In *Web-Age Information Management*, 37–48. doi:10.1007/978-3-642-16720-1\_4. (Cited on page 42).
- Dryad. 2014. *Dryad Digital Depository website*. Accessed 12th January 2014. <http://datadryad.org/>. (Cited on page 43).
- Dublin Core Metadata Initiative. 2004. *Dublin Core Metadata Element Set, version 1.1: Reference description*. Accessed 9th August 2013. <http://dublincore.org/documents/dces/>. (Cited on page 18).
- Eddings, David. 1987. *Guardians of the west*. 7th August. ISBN: 978-0-552-14802-3. (Cited on page v).
- Eijndhoven, Kwame van, and Maurits van der Graaf. 2007. *Inventory study into the present type and level of OAI compliant digital repository activities in the EU*. White Paper OZ 07.1301. DRIVER project, March. (Cited on page 5).
- Ekker, Neal, Tom Coughlin and Jim Handy. 2009. *Solid state storage 101 – An introduction to solid state storage*. Solid State Storage Initiative, January. (Cited on page 19).
- EMBL Outstation – The European Bioinformatics Institute. 2014. *European Nucleotide Archive annotated/assembled sequences: Release Notes*. Version 120, June. Accessed 19th August 2014. <ftp://ftp.ebi.ac.uk/pub/databases/embl/release/relnotes.txt>. (Cited on page 49).
- Etzold, T., A. Ulyanov and P. Argos. 1996. 'SRS: information retrieval system for molecular biology data banks'. *Methods in Enzymology* 266:114–28. (Cited on page 49).
- Evans, Lyndon, and Philip Bryant. 2008. 'LHC machine'. *Journal of Instrumentation* 3, no. 08 (August): S08001. doi:10.1088/1748-0221/3/08/S08001. (Cited on page 49).

- Fallside, David C., and Priscilla Walmsley, eds. 2004. *XML Schema part 0: Primer second edition*. W3C Recommendation. World Wide Web Consortium, 28th October. Accessed 29th April 2014. <http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/>. (Cited on page 13).
- Fielding, R., J. Gettys, J. Mogul, H. Frystyk and T. Berners-Lee. 1997. *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2068 (Proposed Standard). Obsoleted by RFC 2616. Internet Engineering Task Force, January. Accessed 27th March 2014. <http://www.ietf.org/rfc/rfc2068.txt>. (Cited on page 44).
- Fielding, Roy Thomas. 2000. ‘Architectural styles and the design of network-based software architectures’. PhD diss., University of California, Irvine. (Cited on page 55).
- Fiore, Sandro, Alessandro Negro and Giovanni Aloisio. 2011. ‘The data access layer in the GRelC system architecture’. *Future Generation Computer Systems* 27, no. 3 (March): 334–40. ISSN: 0167739X. doi:10.1016/j.future.2010.07.006. (Cited on page 47).
- Flanders, David, Tim Austin, Steven Johnston, Philippa A. Reed, Kenji Takeda and Mark Scott. 2009. EP2DC project meeting with JISC programme manager, 29th September. Faculty of Engineering and the Environment, University of Southampton, United Kingdom. (Cited on page 85).
- Foster, Ian, Carl Kesselman, Jeffrey M. Nick and Steven Tuecke. 2003. ‘The physiology of the Grid’. In *Grid Computing: Making the Global Infrastructure a Reality*, edited by Fran Berman, Geoffrey Fox and Tony Hey, 217–49. Chichester, UK: John Wiley & Sons. ISBN: 978-0-470-86716-7. doi:10.1002/0470867167.ch8. (Cited on page 47).
- Fox, Peter, and Ray Harris. 2013. ‘ICSU and the challenges of data and information management for international science’. In ‘Proceedings of the 1st WDS Conference in Kyoto 2011’, *Data Science Journal* 12 (10th February): WDS1–WDS12. doi:10.2481/dsj.WDS-001. (Cited on page 2).
- Frey, Jeremy. 2008. ‘Curation of laboratory experimental data as part of the overall data lifecycle’. *International Journal of Digital Curation* 3 (1): 44–62. ISSN: 1746-8256. doi:10.2218/ijdc.v3i1.41. (Cited on page 47).
- Gabb, Timothy P., Rebecca A. MacKay, Susan L. Draper, Chantal K. Sudbrack and Michael V. Nathal. 2013. *The mechanical properties of candidate superalloys for a hybrid turbine disk*. Technical report TM-2013-217901. NASA, 1st July. (Cited on page 76).
- Gandon, Fabien, and Guus Schreiber, eds. 2014. *RDF 1.1 XML syntax*. W3C Recommendation. World Wide Web Consortium, 25th February. Accessed 29th April 2014. <http://www.w3.org/TR/2014/REC-rdf-syntax-grammar-20140225/>. (Cited on page 44).

- Gantz, John F., Christopher Chute, Alex Manfrediz, Stephen Minton, David Reinsel, Wolfgang Schlichting and Anna Toncheva. 2008. *The diverse and exploding Digital Universe: An updated forecast of worldwide information growth through 2011*. White Paper. IDC Sponsored by EMC Corporation, March. (Cited on page 2).
- Gantz, John F., and David Reinsel. 2011. *Extracting value from chaos*. White Paper. IDC Sponsored by EMC Corporation, June. (Cited on page 2).
- Gantz, John, and David Reinsel. 2012. *The Digital Universe in 2020: Big data, bigger digital shadows, and biggest growth in the Far East*. White Paper. IDC sponsored by EMC Corporation, December. (Cited on page 2).
- Girone, M. 2008. ‘CERN database services for the LHC computing grid’. *Journal of Physics: Conference Series* 119, no. 5 (July): 052017. ISSN: 1742-6596. doi:10.1088/1742-6596/119/5/052017. (Cited on page 50).
- Goldberg, Ilya G., Chris Allan, Jean-Marie Burel, Doug Creager, Andrea Falconi, Harry Hochheiser, Josiah Johnston, Jeff Mellen, Peter K. Sorger and Jason R. Swedlow. 2005. ‘The Open Microscopy Environment (OME) Data Model and XML file: Open tools for informatics and quantitative analysis in biological imaging’. *Genome biology* 6, no. 5 (May): R47. ISSN: 1465-6914. doi:10.1186/gb-2005-6-5-r47. (Cited on page 48).
- Google. 2010. *Google Storage for developers website*. Accessed 27th March 2014. <http://code.google.com/apis/storage/>. (Cited on page 22).
- Greenberg, Albert, James Hamilton, David A. Maltz and Parveen Patel. 2009. ‘The cost of a cloud: Research problems in data center networks’. *ACM SIGCOMM Computer Communication Review* 39, no. 1 (January): 68–73. ISSN: 0146-4833. doi:10.1145/1496091.1496103. (Cited on page 22).
- Greengard, Samuel. 2013. ‘A new approach to information storage’. *Communications of the ACM* 56, no. 8 (August): 13–15. ISSN: 0001-0782. doi:10.1145/2492007.2492013. (Cited on page 20).
- Gregorio, J., and B. de hOra. 2007. *The Atom Publishing Protocol*. RFC 5023 (Proposed Standard). Internet Engineering Task Force, October. Accessed 27th March 2014. <http://www.ietf.org/rfc/rfc5023.txt>. (Cited on page 44).
- Group 47. 2014. *DOTS website*. Accessed 27th March 2014. <http://www.group47.com/>. (Cited on page 20).
- Guha, Ramanathan V., and Dan Brickley. 2004. *RDF vocabulary description language 1.0: RDF Schema*. W3C Recommendation. World Wide Web Consortium, February. Accessed 2nd September 2011. <http://www.w3.org/TR/2004/REC-rdf-schema-20040210/>. (Cited on page 44).
- Gutteridge, Christopher. 2002. ‘GNU EPrints 2 overview’. In *11th Panhellenic Academic Libraries Conference*. (Cited on page 42).

- Gutteridge, Christopher. 2010. 'Using the institutional repository to publish research data'. In *Proceedings of the 5th Open Knowledge Conference: OKCon 2010*, 1–7. London, UK, 24th April. (Cited on page 43).
- Hawkins, Eleanor, and Mark Scott. 2012. User feedback meeting, 14th December. Faculty of Engineering and the Environment, University of Southampton, United Kingdom. (Cited on page 93).
- Heimbigner, Dennis, and Dennis McLeod. 1985. 'A federated architecture for information management'. *ACM Transactions on Information Systems* 3, no. 3 (July): 253–78. ISSN: 1046-8188. doi:10.1145/4229.4233. (Cited on page 40).
- Hey, Tony, and Anne Trefethen. 2003. 'The Data Deluge: An e-Science perspective'. Chap. 36 in *Grid Computing – Making the Global Infrastructure a Reality*, 809–24. John Wiley & Sons. doi:10.1002/0470867167.ch36. (Cited on page 2).
- Higgins, Sarah. 2008. 'The DCC curation lifecycle model'. *The International Journal of Digital Curation* 3 (1): 134–40. ISSN: 1746-8256. doi:10.2218/ijdc.v3i1.48. (Cited on pages 3, 4).
- Hilbert, Martin, and Priscila López. 2011. 'The world's technological capacity to store, communicate, and compute information'. *Science* 332, no. 6025 (1st April): 60–65. doi:10.1126/science.1200970. (Cited on page 2).
- House of Commons Science and Technology Committee. 2010. *The disclosure of climate data from the Climatic Research Unit at the University of East Anglia*. Technical report HC 387-I. House of Commons, March. (Cited on page 3).
- Howard, John H., Michael L. Kazar, Sherri G. Menees, David A. Nichols, Mahadev Satyanarayanan, Robert N. Sidebotham and Michael J. West. 1988. 'Scale and performance in a distributed file system'. *ACM Transactions on Computer Systems* 6, no. 1 (February): 51–81. doi:10.1145/35037.35059. (Cited on page 22).
- Hubmann-Haidvogel, Alexander, Arno Scharl and Albert Weichselbraun. 2009. 'Multiple coordinated views for searching and navigating web content repositories'. In 'Web Search', *Information Sciences* 179, no. 12 (30th May): 1813–21. doi:10.1016/j.ins.2009.01.030. (Cited on page 41).
- Humphrey, Charles. 2006. *e-Science and the life cycle of research*. Accessed 20th February 2012. <http://datalib.library.ualberta.ca/~humphrey/lifecycle-science060308.doc>. (Cited on pages 155, 156).
- IBM Corporation. 1995. *Character data representation architecture: Reference and registry*. Technical report SC09-2190-00. IBM Corporation, November. (Cited on page 13).
- ITU Telecommunication Development Bureau. 2013. *World in 2013: ICT facts and figures*. Report. International Telecommunication Union (ITU). (Cited on page 2).

- Jäger, Margus, Liina Kamm, Darja Krushevskaja, Harry-Anton Talvik, Janno Veldemann, Andres Vilgota and Jaak Vilo. 2009. 'Flexible database platform for biomedical research with multiple user interfaces and a universal query engine'. In *Databases and Information Systems V: Selected Papers from the Eighth International Baltic Conference, DB&IS 2008*, edited by Hele-Mai Haav and Ahto Kalja, 187:301–10. Frontiers in Artificial Intelligence and Applications. IOS Press, January. ISBN: 978-1-58603-939-4. doi:10.3233/978-1-58603-939-4-301. (Cited on page 47).
- Jiang, R., S. Everitt, M. Lewandowski, N. Gao and P. A. S. Reed. 2013. 'Grain size effects in a Ni-based turbine disc alloy in the time and cycle dependent crack growth regimes'. In '9th Fatigue Damage of Structural Materials Conference', *International Journal of Fatigue* 62 (May): 217–27. ISSN: 0142-1123. doi:10.1016/j.ijfatigue.2013.07.014. (Cited on page 76).
- John of Salisbury. 1955. *The Metalogicon of John of Salisbury: A twelfth-century defense of the verbal and logical arts of the trivium*. Edited by Daniel D. MacGarry. University of California Press. (Cited on page 2).
- Johnston, Steven, Hans Fangohr and Simon J. Cox. 2008. 'Managing large volumes of distributed scientific data'. In *Computational Science – ICCS 2008: 8th International Conference, Proceedings, Part III*, edited by Marian Bubak, Geert Dick van Albada, Jack Dongarra and Peter M. A. Sloot, 5103:339–48. Lecture Notes in Computer Science. Kraków, Poland: Springer Berlin Heidelberg, 23rd–25th June. ISBN: 978-3-540-69388-8. doi:10.1007/978-3-540-69389-5\_39. (Cited on page 2).
- Jones, Richard, Stuart Lewis, Julie Allinson, Tim Brody, David Flanders, Graham Klyne, Alister Miles et al. 2012. *SWORD 2.0 Profile*. Accessed 29th April 2014. <http://swordapp.github.com/SWORDv2-Profile/SWORDProfile.html>. (Cited on page 44).
- Jones, Sarah. 2012. 'Developments in research funder data policy'. *The International Journal of Digital Curation* 7 (1): 114–25. ISSN: 1746-8256. doi:10.2218/ijdc.v7i1.219. (Cited on page 3).
- Kak, Aninash C., and Malcolm Slaney. 2001. *Principles of computerized tomographic imaging*. Society for Industrial and Applied Mathematics. ISBN: 978-0-89871-494-4. (Cited on pages 29, 34).
- Kaminuma, Eli, Takehide Kosuge, Yuichi Kodama, Hideo Aono, Jun Mashima, Takashi Gojobori, Hideaki Sugawara et al. 2011. 'DDBJ progress report'. *Nucleic Acids Research* 39, no. Database issue (January): D22–D27. ISSN: 1362-4962. doi:10.1093/nar/gkq1041. (Cited on page 49).
- Karwin, B. 2010. *SQL antipatterns: Avoiding the pitfalls of database programming*. Pragmatic Bookshelf, 5th July. ISBN: 978-1-934356-55-5. (Cited on pages 37, 38).
- Khurshudov, Andrei. 2001. *The essential guide to computer data storage: From floppy to DVD*. Prentice Hall, 7th June. ISBN: 978-0-13-092739-2. (Cited on page 19).

- Kulikova, Tamara, Philippe Aldebert, Nicola Althorpe, Wendy Baker, Kirsty Bates, Paul Browne, Alexandra van den Broek et al. 2004. 'The EMBL nucleotide sequence database'. *Nucleic Acids Research* 32, no. suppl 1 (January): D27–D30. ISSN: 1362-4962. doi:10.1093/nar/gkh120. (Cited on page 49).
- Kumar, A., HoonJae Lee and R. P. Singh. 2012. 'Efficient and secure cloud storage for handling big data'. In *Proceedings of the 6th International Conference on New Trends in Information Science and Service Science and Data Mining: ISSDM*, 162–66. Tapei, 23rd–25th October. ISBN: 978-1-4673-0876-2. (Cited on page 22).
- Lagoze, Carl, Herbert Van de Sompel, Pete Johnston, Michael Nelson, Robert Sanderson and Simeon Warner, eds. 2008. *ORE user guide – Primer*. Open Archives Initiative, 17th October. Accessed 25th May 2014. <http://www.openarchives.org/ore/1.0/primer>. (Cited on page 44).
- Lagoze, Carl, Herbert Van de Sompel, Michael Nelson and Simeon Warner, eds. 2002. *The Open Archives Initiative Protocol for Metadata Harvesting*. Open Archives Initiative, 14th June. Accessed 25th May 2014. <http://www.openarchives.org/OAI/2.0/openarchivesprotocol.htm>. (Cited on page 45).
- Lassila, Ora, and Ralph R. Swick. 1999. *Resource Description Framework (RDF) model and syntax specification*. W3C Recommendation. World Wide Web Consortium, 22nd February. Accessed 27th March 2014. <http://www.w3.org/TR/1999/REC-rdf-syntax-19990222>. (Cited on page 44).
- Lee, F. 1995. *HZ - A data format for exchanging files of arbitrarily mixed Chinese and ASCII characters*. RFC 1843 (Informational). Internet Engineering Task Force, August. <http://www.ietf.org/rfc/rfc1843.txt>. (Cited on page 15).
- Lee, S. Y., Y. L. Lu, P. K. Liaw, L. J. Chen, S. A. Thompson, J. W. Blust, P. F. Browning, A. K. Bhattacharya, J. M. Aurrecoechea and D. L. Klarstrom. 2009. 'Tensile-hold low-cycle-fatigue properties of solid-solution-strengthened superalloys at elevated temperatures'. *Materials Science and Engineering: A* 504, nos. 1–2 (25th March): 64–72. ISSN: 0921-5093. doi:10.1016/j.msea.2008.10.030. (Cited on page 25).
- Leiner, Barry M., Vinton G. Cerf, David D. Clark, Robert E. Kahn, Leonard Kleinrock, Daniel C. Lynch, Jon Postel, Larry G. Roberts and Stephen Wolff. 2009. 'A brief history of the Internet'. *ACM SIGCOMM Computer Communication Review* 39, no. 5 (October): 22–31. doi:10.1145/1629607.1629613. (Cited on page 22).
- Library of Congress – Network Development and MARC Standards Office. 2013a. *MARC Standards website*. Accessed 9th August 2013. <http://www.loc.gov/marc/>. (Cited on page 18).
- . 2013b. *Metadata Object Description Schema website*. Accessed 9th August 2013. <http://www.loc.gov/standards/mods/>. (Cited on page 18).

- Linden, G., B. Smith and J. York. 2003. 'Amazon.com recommendations: Item-to-item collaborative filtering'. *Internet Computing, IEEE* 7 (1): 76–80. ISSN: 1089-7801. doi:10.1109/MIC.2003.1167344. (Cited on page 85).
- Linkert, Melissa, Curtis T. Rueden, Chris Allan, Jean-Marie Burel, Will Moore, Andrew Patterson, Brian Loranger et al. 2010. 'Metadata matters: Access to image data in the real world'. *The Journal of Cell Biology* 189, no. 5 (May): 777–82. ISSN: 1540-8140. doi:10.1083/jcb.201004104. (Cited on pages 47, 48).
- Llorca, J., A. Martin, J. Ruiz and M. Elices. 1993. 'Particulate fracture during deformation'. *Metallurgical Transactions A* 24, no. 7 (July): 1575–88. ISSN: 1073-5623. doi:10.1007/BF02646597. (Cited on page 28).
- Love, Robert. 2005. 'Kernel Korner: Intro to inotify'. *Linux Journal* (Houston, TX) 2005, no. 139 (November). ISSN: 1075-3583. (Cited on page 151).
- Lunt, Barry M., Matthew R. Linford, Robert Davis, Hao Wang, Douglas Hansen and John Dredge. 2013. 'Permanent digital data storage: A materials approach'. In *Proceedings of the 10th International Conference on Preservation of Digital Objects: iPRES 2013*, 209–14. Lisbon, Portugal, 3rd–5th September. ISBN: 978-972-565-493-4. (Cited on page 20).
- Marenco, Luis, Nicholas Tosches, Chiquito Crasto, Gordon Shepherd, Perry L. Miller and Prakash M. Nadkarni. 2003. 'Achieving evolvable web-database bioscience applications using the EAV/CR framework: Recent advances'. *Journal of the American Medical Informatics Association* 10 (5): 444–53. ISSN: 1067-5027. doi:10.1197/jamia.M1303. (Cited on page 47).
- Marinai, S. 2009. 'Metadata extraction from PDF papers for digital library ingest'. In *Proceedings of the 10th International Conference on Document Analysis and Recognition: ICDAR '09*, 251–55. Barcelona, Spain, 26th July–29th July. doi:10.1109/ICDAR.2009.232. (Cited on page 18).
- Marsh, B., F. Douglass and P. Krishnan. 1994. 'Flash memory file caching for mobile computers'. In *Proceedings of the 27th Hawaii International Conference on System Sciences*, 1:451–60. Wailea, HI, USA, 4th January–7th January. ISBN: 978-0-8186-5090-1. doi:10.1109/HICSS.1994.323153. (Cited on page 20).
- Masters, Tim. 2013. 'Doctor Who: Yeti classic among episodes found in Nigeria'. *BBC News* (11th October). <http://www.bbc.co.uk/news/entertainment-arts-24467337>. (Cited on page 1).
- Mbuya, Thomas O., Ian Sinclair, Andrew J. Moffat and Philippa A. S. Reed. 2011. 'Analysis of fatigue crack initiation and S-N response of model cast aluminium piston alloys'. *Materials Science and Engineering: A* 528, no. 24 (15th September): 7331–40. ISSN: 0921-5093. doi:10.1016/j.msea.2011.06.007. (Cited on page 25).



- McKenna, Aaron, Matthew Hanna, Eric Banks, Andrey Sivachenko, Kristian Cibulskis, Andrew Kernysky, Kiran Garimella et al. 2010. 'The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data'. *Genome Research* 20 (9): 1297–303. doi:10.1101/gr.107524.110. (Cited on page 103).
- Meijer, Erik, and Gavin Bierman. 2011. 'A co-relational model of data for large shared data banks'. *Communications of the ACM* 54, no. 4 (April): 49–58. ISSN: 00010782. doi:10.1145/1924421.1924436. (Cited on page 41).
- Merton, R. K. 1965. *On the shoulders of giants: The post-Italianate edition*. University of Chicago Press. ISBN: 978-0-226-52086-5. (Cited on page 2).
- Microsoft. 2001. *Windows XP to take the PC to new heights*, 24th August. Accessed 1st May 2014. <http://www.microsoft.com/en-us/news/press/2001/aug01/08-24winxprrtmpr.aspx>. (Cited on page 152).
- . 2003. *NTFS technical reference: How NTFS works*. 28th March. Accessed 2nd May 2014. [http://technet.microsoft.com/en-us/library/cc781134\(v=ws.10\).aspx](http://technet.microsoft.com/en-us/library/cc781134(v=ws.10).aspx). (Cited on pages 65, 152).
- . 2006. *How to minimize metadata in Word 2003*. Knowledge Base. Accessed 9th February 2014. <http://support.microsoft.com/kb/825576>. (Cited on page 18).
- . 2007. *New capabilities and features of the NTFS 3.1 file system*. Knowledge Base. Version 2.4, 1st December. Accessed 1st May 2014. <http://support.microsoft.com/kb/310749>. (Cited on page 152).
- . 2009a. *STSUpd.UploadCtl cannot handle some exceptions when it posts files to a custom ASPX page*. Knowledge Base. Accessed 11th September 2013. <http://support.microsoft.com/kb/971597>. (Cited on page 102).
- . 2009b. *You cannot download files that are 2 GB or larger*. Knowledge Base. Accessed 2nd September 2011. <http://support.microsoft.com/kb/298618>. (Cited on page 24).
- . 2010a. *FileSystemWatcher class*. December. Accessed 11th May 2014. [http://msdn.microsoft.com/en-us/library/system.io.filesystemwatcher\(v=vs.90\).aspx](http://msdn.microsoft.com/en-us/library/system.io.filesystemwatcher(v=vs.90).aspx). (Cited on page 69).
- . 2010b. *Microsoft Visual Studio 2010 and Microsoft .NET Framework 4 available*, 11th April. Accessed 1st May 2014. <https://www.microsoft.com/en-us/news/press/2010/apr10/04-11vs10pr.aspx>. (Cited on page 152).
- . 2010c. *Windows Azure website*. Accessed 4th June 2010. <http://www.microsoft.com/windowsazure/>. (Cited on page 22).
- . 2013a. *Error message when you visit a web site that is hosted on a server that is running Internet Information Services 7.0: 'HTTP Error 404.13 - CONTENT\_LENGTH\_TOO\_LARGE'*. Knowledge Base. Accessed 11th September 2013. <http://support.microsoft.com/kb/942074>. (Cited on page 102).

- . 2013b. *Microsoft SMB Protocol and CIFS protocol overview*, 12th May. Accessed 9th February 2014. [http://msdn.microsoft.com/en-us/library/windows/desktop/aa365233\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/aa365233(v=vs.85).aspx). (Cited on page 22).
- . 2013c. *SharePoint Server 2010 capacity management: Software boundaries and limits*. 25th June. Accessed 29th April 2014. [http://technet.microsoft.com/en-us/library/cc262787\(v=office.14\).aspx](http://technet.microsoft.com/en-us/library/cc262787(v=office.14).aspx). (Cited on pages 43, 102).
- . 2014a. *Features supported by the editions of SQL Server 2008 R2*. Accessed 2nd May 2014. [http://msdn.microsoft.com/en-us/library/cc645993\(v=SQL.105\).aspx](http://msdn.microsoft.com/en-us/library/cc645993(v=SQL.105).aspx). (Cited on page 152).
- . 2014b. *File systems: Maximum size limitations*. Accessed 2nd May 2014. <http://technet.microsoft.com/en-us/library/cc938937.aspx>. (Cited on pages 65, 152).
- . 2014c. *LINQ (Language-Integrated Query)*. Accessed 26th May 2014. [http://msdn.microsoft.com/en-us/library/bb397926\(v=vs.100\).aspx](http://msdn.microsoft.com/en-us/library/bb397926(v=vs.100).aspx). (Cited on page 90).
- . 2014d. *.NET Framework 3.5 Guid structure*, January. Accessed 9th February 2014. [http://msdn.microsoft.com/en-us/library/system.guid\(v=vs.90\).aspx](http://msdn.microsoft.com/en-us/library/system.guid(v=vs.90).aspx). (Cited on page 57).
- . 2014e. *Software boundaries and limits for SharePoint 2013*. 2nd April. Accessed 25th May 2014. [http://technet.microsoft.com/en-us/library/cc262787\(v=office.15\).aspx](http://technet.microsoft.com/en-us/library/cc262787(v=office.15).aspx). (Cited on page 43).
- . 2014f. *SQL Server 2005 Books Online*. Accessed 4th May 2014. [http://msdn.microsoft.com/en-us/library/ms130214\(v=sql.90\).aspx](http://msdn.microsoft.com/en-us/library/ms130214(v=sql.90).aspx). (Cited on page 65).
- Midgard Project. 2011. *The Midgard Project website*. Accessed 19th July 2011. <http://www.midgard-project.org/>. (Cited on page 42).
- Min, Changwoo, Sang-Won Lee and Young Ik Eom. 2013. 'Design and implementation of a log-structured file system for flash-based solid state drives'. *IEEE Transactions on Computers* 63 (9): 2215–27. ISSN: 0018-9340. doi:10.1109/TC.2013.97. (Cited on page 21).
- Mizonov, Valery, and Seth Manheim. 2013. *Windows Azure table storage and Windows Azure SQL database – Compared and contrasted*. Accessed 31st October 2013. <http://msdn.microsoft.com/en-us/library/windowsazure/jj553018.aspx>. (Cited on page 51).
- Moen, William E., and Penelope Benardino. 2003. 'Assessing metadata utilization: An analysis of MARC content designation use'. In *Proceedings of the 2003 international conference on Dublin Core and metadata applications: Supporting communities of discourse and practice – metadata research & applications: DCMI '03*, 171–80. Seattle, Washington: Dublin Core Metadata Initiative, 28th September–2nd October. ISBN: 978-0-9745303-0-7. (Cited on page 18).

- Mwebaze, J., D. Boxhoorn and E. Valentijn. 2009. 'Astro-WISE: Tracing and using lineage for scientific data processing'. In *International Conference on Network-Based Information Systems: NBIS '09*, 475–80. 19th–21st August. doi:10.1109/NBiS.2009.48. (Cited on page 47).
- MySQL Documentation Team. 2011. *MySQL 5.6 reference manual*. Oracle Corporation, 19th November. Accessed 22nd November 2011. <http://dev.mysql.com/doc/refman/5.6/en/>. (Cited on page 51).
- Nadkarni, Prakash M., Luis Marenco, Roland Chen, Emmanouil Skoufos, Gordon Shepherd and Perry Miller. 1999. 'Organization of heterogeneous scientific data using the EAV/CR representation'. *Journal of the American Medical Informatics Association* 6 (6): 478–93. ISSN: 1067-5027. doi:10.1136/jamia.1999.0060478. (Cited on pages 38, 47).
- National Institute of Standards and Technology and U.S. Department of Commerce. 2012. *FIPS PUB 180-4: Secure Hash Standard (SHS)*. Federal Information Processing Standards Publication 180-4. March. (Cited on page 24).
- National Science and Technology Council. 2011. *Materials Genome Initiative for global competitiveness*. White Paper, June. Accessed 10th February 2014. <http://www.whitehouse.gov/mgi>. (Cited on page 5).
- Nature. 2013. *Nature guide to authors: Final submissions*. PDF, 13th August. Accessed 10th February 2014. <http://www.nature.com/nature/authors/submissions>. (Cited on page 3).
- Newton, Isaac. 1959. *The correspondence of Isaac Newton*. Edited by H. W. Turnbull. Vol. 1. Cambridge University Press for the Royal Society. (Cited on page 2).
- Ng, Muan Hong, Steven Johnston, Bing Wu, Stuart E. Murdock, Kaihsu Tai, Hans Fangohr, Simon J. Cox, Jonathan W. Essex, Mark S. P. Sansom and Paul Jeffreys. 2006. 'BioSimGrid: grid-enabled biomolecular simulation data storage and analysis'. *Future Generation Computer Systems* 22 (6): 657–64. doi:10.1016/j.future.2005.10.005. (Cited on page 47).
- Novocraft. 2013. *Novoalign website*, September. Accessed 12th September 2013. <http://www.novocraft.com>. (Cited on page 103).
- Novotny, Eric. 2004. 'I don't think I click: A protocol analysis study of use of a library online catalog in the Internet age'. *College & Research Libraries* 65, no. 6 (1st November): 525–37. ISSN: 0010-0870. (Cited on page 45).
- Nuescheler, David. 2005. *JSR-000170 Content Repository for Java technology API 1.0 Final Release*, June. Accessed 19th July 2011. <http://www.jcp.org/en/jsr/detail?id=170>. (Cited on page 42).

- . 2009. *JSR-000283 Content Repository for Java technology API 2.0 Final Release*, September. Accessed 19th July 2011. <http://www.jcp.org/en/jsr/detail?id=283>. (Cited on page 42).
- Ojala, Tauno, and Hans-Helmut Over. 2008. 'Approaches in using MatML as a common language for materials data exchange'. *Data Science Journal* 7 (November): 179–95. ISSN: 1683-1470. doi:10.2481/dsj.7.179. (Cited on pages 5, 30, 161).
- OpenAFS Project. 2014. *OpenAFS website*. Accessed 9th February 2014. <https://www.openafs.org>. (Cited on page 22).
- Oxford English Dictionary. 2012. 'data, n.' In *OED Online*. Oxford University Press, December. Accessed 5th May 2013. <http://www.oed.com/view/Entry/296948>. (Cited on page 11).
- Pääbo, Svante, Hendrik Poinar, David Serre, Viviane Jaenicke-Després, Juliane Hebler, Nadin Rohland, Melanie Kuch, Johannes Krause, Linda Vigilant and Michael Hofreiter. 2004. 'Genetic analyses from ancient DNA'. *Annual Review of Genetics* 38 (December): 645–79. doi:10.1146/annurev.genet.37.110801.143214. (Cited on page 20).
- Palankar, Mayur R., Adriana Iamnitchi, Matei Ripeanu and Simson Garfinkel. 2008. 'Amazon S3 for science grids: A viable solution?' In *Proceedings of the 2008 International Workshop on Data-aware Distributed Computing: DADC '08*, 55–64. Boston, MA, USA: ACM, 23rd June–27th July. ISBN: 978-1-60558-154-5. doi:10.1145/1383519.1383526. (Cited on page 22).
- Pallickara, Sangmi Lee, Shrideep Pallickara and Milija Zupanski. 2012. 'Towards efficient data search and subsetting of large-scale atmospheric datasets'. *Future Generation Computer Systems* 28 (1): 112–18. ISSN: 0167-739X. doi:10.1016/j.future.2011.05.010. (Cited on page 48).
- Pang, H. T., and P. A. S. Reed. 2009. 'Microstructure variation effects on room temperature fatigue threshold and crack propagation in Udimet 720Li Ni-base superalloy'. *Fatigue & Fracture of Engineering Materials & Structures* 32, no. 8 (August): 685–701. ISSN: 1460-2695. doi:10.1111/j.1460-2695.2009.01366.x. (Cited on page 25).
- Parrington, Ronald J. 2002. 'Fractography of metals and plastics'. *Practical Failure Analysis* 2, no. 5 (October): 16–19. ISSN: 1529-8159. doi:10.1007/BF02715463. (Cited on page 28).
- Pawley, E. L. E. 1961. 'B.B.C. television 1939-60. A review of progress'. *Proceedings of the IEE – Part B: Electronic and Communication Engineering* 108, no. 40 (July): 375–97. ISSN: 0369-8890. doi:10.1049/pi-b-2.1961.0066. (Cited on page 1).

- Payette, Sandra, and Carl Lagoze. 1998. 'Flexible and Extensible Digital Object and Repository Architecture (FEDORA)'. In *Proceedings of the 2nd European Conference on Research and Advanced Technology for Digital Libraries: ECDL '98*, 1513:41–59. Lecture Notes in Computer Science. Heraklion, Crete, Greece, 21st September–23rd September. ISBN: 978-3-540-65101-7. doi:10.1007/3-540-49653-X\_4. (Cited on page 42).
- Peters, Andreas J., P. Saiz and P. Buncic. 2003. 'AliEnFS – a Linux file system for the AliEn Grid services'. In *Proceedings of the Conference for Computing in High Energy and Nuclear Physics: CHEP '03*. La Jolla, California, 24th March–28th March. arXiv: cs/0306071. (Cited on page 50).
- Pinheiro, Eduardo, Wolf-Dietrich Weber and Luiz André Barroso. 2007. 'Failure trends in a large disk drive population'. In *Proceedings of the 5th USENIX Conference on File and Storage Technologies: FAST '07*, 17–28. San Jose, CA: USENIX Association, 14th February–16th February. (Cited on page 19).
- Potter, Simon C., Laura Clarke, Val Curwen, Stephen Keenan, Emmanuel Mongin, Stephen M. J. Searle, Arne Stabenau, Roy Storey and Michele Clamp. 2004. 'The Ensembl analysis pipeline'. *Genome research* 14, no. 5 (May): 934–41. ISSN: 1088-9051. doi:10.1101/gr.1859804. (Cited on page 49).
- Pouwelse, Johan, Paweł Garbacki, Dick Epema and Henk Sips. 2005. 'The Bittorrent P2P file-sharing system: Measurements and analysis'. In *Peer-to-Peer Systems IV*, edited by Miguel Castro and Robbert Renesse, 3640:205–16. Lecture Notes in Computer Science. Springer Berlin Heidelberg. ISBN: 978-3-540-29068-1. doi:10.1007/11558989\_19. (Cited on page 23).
- Prabhakaran, Vijayan, Andrea C. Arpaci-Dusseau and Remzi H. Arpaci-Dusseau. 2005. 'Analysis and evolution of journaling file systems'. In *Proceedings of the USENIX Annual Technical Conference: ATEC '05*, 105–20. Anaheim, CA, USA, 10th April–15th April. (Cited on page 21).
- Pringle, Ashley. 1971. 'TV studies: An introduction to a critique of television'. *Screen* 12 (1): 63–71. ISSN: 0036-9543. doi:10.1093/screen/12.1.63. (Cited on page 1).
- Prud'hommeaux, Eric, and Andy Seaborne, eds. 2008. *SPARQL Query Language for RDF*. W3C Recommendation, 15th January. Accessed 27th March 2014. <http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/>. (Cited on pages 44, 102).
- Raman, Karthik V., Alexander M. Kamerbeek, Arup Mukherjee, Nicolae Atodiresei, Tamal K. Sen, Predrag Lazić, Vasile Caciuc et al. 2013. 'Interface-engineered templates for molecular spin memory devices'. *Nature* 493, no. 7433 (24th January): 509–13. doi:10.1038/nature11719. (Cited on page 20).

- Reed, P. A. S., and J. F. Knott. 1996. ‘Investigation of the role of residual stresses in the warm prestress (WPS) effect. Part II – Analysis’. *Fatigue & Fracture of Engineering Materials & Structures* 19, no. 4 (April): 501–13. ISSN: 1460-2695.  
doi:10.1111/j.1460-2695.1996.tb00985.x. (Cited on page 25).
- Reed, Philippa A., and Mark Scott. 2011. User feedback meeting, 18th January. Faculty of Engineering and the Environment, University of Southampton, United Kingdom. (Cited on pages 93, 102).
- . 2012. User feedback meeting, 16th November. Faculty of Engineering and the Environment, University of Southampton, United Kingdom. (Cited on page 85).
- Reed, Philippa A., and Kath Soady. 2010. Interview by Mark Scott, 26th April. Faculty of Engineering and the Environment, University of Southampton, United Kingdom. (Cited on pages 30, 75, 93).
- Reed, Philippa, Tim Austin and Kenji Takeda. 2009. *Materials Data Centre*. Project Proposal to JISC, April. (Cited on page 11).
- Reed, Philippa, Leslie Carr, Kenji Takeda and Tim Austin. 2009. *EP2MDC*. Project Proposal to JISC, June. Accessed 9th February 2014. <https://jiscr.googlecode.com/files/74%20University%20of%20Southampton%20EP2MDC.pdf>. (Cited on pages 11, 53).
- Research Information Network. 2008. *Stewardship of digital research data: A framework of principles and guidelines*, January. Accessed 20th February 2012. <http://www.rin.ac.uk/our-work/data-management-and-curation/stewardship-digital-research-data-principles-and-guidelines>. (Cited on page 116).
- Rey, Alain. 2012. ‘The Nanoform: Sapphire disk system of records preservation used by ANDRA’. In *The Preservation of Records, Knowledge and Memory (RK&M) Across Generations: Improving Our Understanding: Workshop Proceedings*, 81–85. Issy-les-Moulineaux, France: OECD/NEA Radioactive Waste Management Committee, 12th September–13th September. (Cited on page 20).
- Rimer, Charlotte, and Philippa Reed. 2009. *Materials Data Centre – Report on user requirements*. Technical report. University of Southampton, August. (Cited on pages 5, 6, 11, 24, 25, 157).
- Rivest, R. 1992. *The MD5 message-digest algorithm*. RFC 1321 (Informational). Internet Engineering Task Force, April. Accessed 26th May 2014. <http://www.ietf.org/rfc/rfc1321.txt>. (Cited on page 24).
- Rodeh, Ohad, Josef Bacik and Chris Mason. 2013. ‘BTRFS: The Linux B-Tree filesystem’. *ACM Transactions on Storage* 9, no. 3 (August): 9:1–9:32. ISSN: 1553-3077.  
doi:10.1145/2501620.2501623. (Cited on page 21).

- Rodrigues, Rodrigo, and Peter Druschel. 2010. 'Peer-to-peer systems'. *Communications of the ACM* 53, no. 10 (October): 72–82. ISSN: 0001-0782. doi:10.1145/1831407.1831427. (Cited on page 23).
- Rosenblum, Mendel, and John K Ousterhout. 1992. 'The design and implementation of a log-structured file system'. *ACM Transactions on Computer Systems* 10, no. 1 (February): 26–52. doi:10.1145/146941.146943. (Cited on page 21).
- Ross, C. A. 2001. 'Patterned magnetic recording media'. *Annual Review of Materials Research* 31, no. 1 (August): 203–35. doi:10.1146/annurev.matsci.31.1.203. (Cited on page 20).
- Ruffalo, Matthew, Thomas LaFramboise and Mehmet Koyutürk. 2011. 'Comparative analysis of algorithms for next-generation sequencing read alignment'. *Bioinformatics* 27, no. 20 (19th August): 2790–96. doi:10.1093/bioinformatics/btr477. (Cited on page 103).
- Russinovich, Mark. 2000. 'Inside Win2K NTFS, Part 1'. *Windows IT Pro* (22nd October). (Cited on page 64).
- Rys, Michael. 2011. 'Scalable SQL'. *Communications of the ACM* 54, no. 6 (June). ISSN: 00010782. doi:10.1145/1953122.1953141. (Cited on page 41).
- Sallans, Andrew, and Martin Donnelly. 2012. 'DMP Online and DMPTool: Different strategies towards a shared goal'. *The International Journal of Digital Curation* 7 (2): 123–29. ISSN: 1746-8256. doi:10.2218/ijdc.v7i2.235. (Cited on page 4).
- Sandberg, Russel, David Goldberg, Steve Kleiman, Dan Walsh and Bob Lyon. 1985. 'Design and implementation of the Sun network filesystem'. In *Proceedings of the Summer USENIX conference*, 119–30. Portland, Oregon, USA, 11th–14th June. (Cited on page 22).
- Sandys, John Edwin. 1906. *A history of classical scholarship, from the sixth century BC to the end of the middle ages*. Vol. 1. Cambridge University Press. (Cited on page 2).
- Sargent, R., D. Fuhrman, T. Critchlow, T. Di Sera, R. Mecklenburg, G. Lindstrom and P. Cartwright. 1996. 'The design and implementation of a database for human genome research'. In *Proceedings of the Eighth International Conference on Scientific and Statistical Database Systems*, 220–25. Stockholm: IEEE, 18th June–20th June. ISBN: 978-0-8186-7264-4. doi:10.1109/SSDM.1996.506064. (Cited on page 49).
- Schneider, Michael. 2009. *OWL 2 Web Ontology Language: RDF-based semantics*. W3C Recommendation. World Wide Web Consortium, October. Accessed 2nd September 2011. <http://www.w3.org/TR/2009/REC-owl2-rdf-based-semantics-20091027/>. (Cited on page 44).

- Schroeder, Bianca, and Garth A. Gibson. 2007. 'Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you?' In *Proceedings of the 5th USENIX Conference on File and Storage Technologies: FAST '07*, 1–16. San Jose, CA: USENIX Association, 14th February–16th February. (Cited on page 19).
- Scott, Mark, Richard P. Boardman, Philippa A. Reed, Tim Austin, Steven J. Johnston, Kenji Takeda and Simon J. Cox. 2014. 'A framework for user driven data management'. *Information Systems* 42 (June): 36–58. doi:10.1016/j.is.2013.11.004. (Cited on pages xix, 9, 11, 63, 153).
- Scott, Mark, Richard P. Boardman, Philippa A. Reed, Dorothy Byatt and Simon J. Cox. 2013. 'Research data management education for future curators'. In *8th International Digital Curation Conference*. Amsterdam, Netherlands: The Digital Curation Centre, 14th January. (Cited on pages xx, 9, 115, 154).
- Scott, Mark, Richard P. Boardman, Philippa A. Reed and Simon J. Cox. 2013a. 'Introducing research data'. Edited by Mark Scott and Dorothy Byatt. JISC DataPool Project, 14th March. Accessed 13th May 2015. <http://eprints.soton.ac.uk/360442/>. (Cited on page 115).
- . 2013b. 'Research data management education for future curators'. *The International Journal of Digital Curation* 8 (1): 288–94. ISSN: 1746-8256. doi:10.2218/ijdc.v8i1.261. (Cited on pages xix, 9, 115, 154).
- . 2014. 'Managing heterogeneous datasets'. *Information Systems* 44 (August): 34–53. doi:10.1016/j.is.2014.03.004. (Cited on pages xix, 9, 95, 154).
- Sears, Russell, Catharine van Ingen and Jim Gray. 2006. *To BLOB or not to BLOB: Large object storage in a database or a filesystem?* Technical report MSR-TR-2006-45. Microsoft Corporation, One Microsoft Way, Redmond, WA 98052: Microsoft Research, April. (Cited on page 65).
- Sha, Guiying, Xiaoguang Sun, Teng Liu, Yuhong Zhu and Tao Yu. 2011. 'Effects of Sc addition and annealing treatment on the microstructure and mechanical properties of the As-rolled Mg-3Li alloy'. *Journal of Materials Science & Technology* 27 (8): 753–58. ISSN: 1005-0302. doi:10.1016/S1005-0302(11)60138-2. (Cited on page 25).
- Simonite, Tom, and Michael Le Page. 2010. 'Digital doomsday: The end of knowledge'. *New Scientist*, no. 2745 (2nd February): 36–39. (Cited on page 19).
- Singh, Amit. 2006. *Mac OS X internals: A systems approach*. Addison-Wesley Professional, 19th June. ISBN: 978-0-321-27854-8. (Cited on page 151).
- Singh, Siddharth Kumar, Michael Witt and Dorothea Salo. 2010. 'A comparative analysis of institutional repository software'. In *Proceedings of the 5th International Conference on Open Repositories: OR2010*. Madrid, Spain, 6th July–9th July. doi:10.2390/biecoll-OR2010-13. (Cited on page 43).



- Smit, Eefke, Jeffrey van der Hoeven and David Giarretta. 2011. ‘Avoiding a Digital Dark Age for data: Why publishers should care about digital preservation’. *Learned Publishing* 24, no. 1 (January): 35–49. ISSN: 09531-513. doi:10.1087/20110107. (Cited on pages 17, 19).
- Soady, Kath, and Mark Scott. 2011. User feedback meeting, 16th December. Faculty of Engineering and the Environment, University of Southampton, United Kingdom. (Cited on page 85).
- Sompel, Herbert Van de, Michael L. Nelson, Carl Lagoze and Simeon Warner. 2004. ‘Resource harvesting within the OAI-PMH framework’. *D-Lib Magazine* 10, no. 12 (December). ISSN: 1082-9873. doi:10.1045/december2004-vandesompel. (Cited on page 45).
- Stead, W. W., W. E. Hammond and M. J. Straube. 1983. ‘A chartless record – Is it adequate?’ *Journal of Medical Systems* 7, no. 2 (April): 103–9. ISSN: 0148-5598. doi:10.1007/BF00995117. (Cited on page 47).
- Stein, Lincoln D., and Jean Thierry-Mieg. 1999. ‘AceDB: A genome database management system’. *Computing in Science & Engineering* 1 (3): 44–52. doi:10.1109/5992.764215. (Cited on page 49).
- SWORD Project. 2009. *SWORD project background*. Accessed 28th April 2014. [http://www.ukoln.ac.uk/repositories/digirep/index/SWORD\\_Project](http://www.ukoln.ac.uk/repositories/digirep/index/SWORD_Project). (Cited on page 44).
- . 2013. *The SWORD website*. Accessed 29th April 2014. <http://swordapp.org>. (Cited on page 44).
- The ALICE Collaboration. 2011. *The ALICE offline bible*. Accessed 19th July 2011. <http://aliceinfo.cern.ch/Offline/AliRoot/Manual.html>. (Cited on page 50).
- The ALICE Collaboration, K. Aamodt, A. Abrahantes Quintana, R. Achenbach, S. Acounis, D. Adamová, C. Adler et al. 2008. ‘The ALICE experiment at the CERN LHC’. In ‘The CERN Large Hadron Collider: Accelerator And Experiments’, *Journal Of Instrumentation* 3, no. 08 (August): S08002. ISSN: 17480221. doi:10.1088/1748-0221/3/08/S08002. (Cited on page 49).
- The Apache Software Foundation. 2011. *Apache Jackrabbit website*. Accessed 19th July 2011. <http://jackrabbit.apache.org>. (Cited on pages 42, 43).
- . 2013. *Apache Derby website*. Accessed 6th December 2013. <http://db.apache.org/derby>. (Cited on page 40).
- The LHCf Collaboration, O. Adriani, L. Bonechi, M. Bongi, G. Castellini, R. D’Alessandro, D. A. Faus et al. 2008. ‘The LHCf detector at the CERN Large Hadron Collider’. In ‘The CERN Large Hadron Collider: Accelerator And Experiments’, *Journal of Instrumentation* 3, no. 08 (August): S08006–S08006. ISSN: 1748-0221. doi:10.1088/1748-0221/3/08/S08006. (Cited on page 49).

- The Minerals, Metals & Materials Society (TMS). 2013. *3D Materials Atlas website*. Accessed 9th June 2013. <https://cosmicweb.mse.iastate.edu/wiki/display/home/Materials+Atlas+Home>. (Cited on page 48).
- The Neo4j Team. 2011. *The Neo4j Manual v1.4*. Accessed 19th July 2011. <http://docs.neo4j.org/chunked/1.4/>. (Cited on page 42).
- The Unicode Consortium. 2012. *The Unicode Standard, version 6.2.0*. Mountain View, CA: The Unicode Consortium. ISBN: 978-1-936213-07-8, accessed 30th September 2014. <http://www.unicode.org/versions/Unicode6.2.0/>. (Cited on page 15).
- Thompson, D. A., and J. S. Best. 2000. 'The future of magnetic data storage technology'. *IBM Journal of Research and Development* 44, no. 3 (May): 311–22. ISSN: 0018-8646. doi:10.1147/rd.443.0311. (Cited on page 19).
- Thompson, Rob. 2010. Personal emails containing material data sheets for U720, N18, U720Li and LSHR, 15th–26th July. Faculty of Engineering and the Environment, University of Southampton, United Kingdom. (Cited on page 35).
- Tinkle, Sally, David L. McDowell, Amanda Barnard, Francois Gygi and Peter B. Littlewood. 2013. 'Technology: Sharing data in materials science'. *Nature* 503 (27th November): 463–64. doi:10.1038/503463a. (Cited on page 5).
- Toledo, J., F. J. Mora and H. Müller. 2003. 'Past, present and future of data acquisition systems in high energy physics experiments'. *Microprocessors and Microsystems* 27, no. 8 (September): 353–58. ISSN: 01419331. doi:10.1016/S0141-9331(03)00065-6. (Cited on page 50).
- Transregional Collaborative Research Centre 32. 2014. *Transregional Collaborative Research Centre 32 Database (TR32DB)*. Accessed 15th January 2014. <http://www.tr32db.uni-koeln.de/site/faq.php>. (Cited on page 48).
- Treloar, Andrew, David Groenewegen and Cathrine Harboe-Ree. 2007. 'The data curation continuum: Managing data objects in institutional repositories'. *D-Lib Magazine* 13, nos. 9/10 (September). ISSN: 1082-9873. doi:10.1045/september2007-treloar. (Cited on page 43).
- Tridgell, Andrew. 1999. 'Efficient algorithms for sorting and synchronization'. PhD diss., Australian National University Canberra. (Cited on page 24).
- University of Dundee & Open Microscopy Environment. 2014. *OME server system overview*. Accessed 13th May 2014. <https://www.openmicroscopy.org/site/support/previous/ome-server/system-overview/>. (Cited on page 48).
- University of Edinburgh. 2011. *Defining research data*, June. Accessed 20th February 2012. <http://www.ed.ac.uk/schools-departments/information-services/services/research-support/data-library/research-data-mgmt/data-mgmt/research-data-definition>. (Cited on page 116).

- University of Southampton Library. 2014. *Research Data Management website*, 5th August. Accessed 7th August 2014. <http://library.soton.ac.uk/researchdata>. (Cited on page 150).
- Wakefield, A. J., S. H. Murch, A. Anthony, J. Linnell, D. M. Casson, M. Malik, M. Berelowitz et al. 1998. 'RETRACTED: Ileal-lymphoid-nodular hyperplasia, non-specific colitis, and pervasive developmental disorder in children'. *The Lancet* 351, no. 9103 (28th February): 637–41. doi:10.1016/S0140-6736(97)11096-0. (Cited on page 3).
- Wang, Xiaogui, Donghui Yin, Feng Xu, Baoxiang Qiu and Zengliang Gao. 2012. 'Fatigue crack initiation and growth of 16MnR steel with stress ratio effects'. *International Journal of Fatigue* 35, no. 1 (February): 10–15. ISSN: 0142-1123. doi:10.1016/j.ijfatigue.2011.05.007. (Cited on page 25).
- White, Wendy, Simon Cox, Graeme Earl, Mark Scott, Dorothy Byatt and Steve Hitchcock. 2013. 'Working collaboratively with PhD and early career researchers: Agents for change'. In *8th International Digital Curation Conference*. Amsterdam, Netherlands: The Digital Curation Centre, 14th January–17th January. (Cited on pages xx, 9, 115).
- Wood, Ken. 2013. 'Optical storage technologies'. In *Proceedings of the 2013 Storage Developers Conference*. Santa Clara, California: Storage Networking Industry Association, 16th–19th September. (Cited on page 20).
- Wood, R. 2000. 'The feasibility of magnetic recording at 1 terabit per square inch'. *IEEE Transactions on Magnetics* 36, no. 1 (January): 36–42. doi:10.1109/20.824422. (Cited on page 20).
- Worsley, Thomas Cuthbert. 1970. *Television: The ephemeral art*. Ross, July. ISBN: 978-0-900626-10-4. (Cited on page 1).
- Wright, Pete, and Anna Scott. 2010. Interview by Mark Scott, 23rd April. Faculty of Engineering and the Environment, University of Southampton, United Kingdom. (Cited on page 28).
- Yeh, Benjamin M., John A. Shepherd, Zhen J. Wang, Hui Seong Teh, Robert P. Hartman and Sven Prevrhal. 2009. 'Dual-energy and low-kVp CT in the abdomen'. *American Journal of Roentgenology* 193, no. 1 (1st July): 47–54. ISSN: 0361-803X. doi:10.2214/AJR.09.2592. (Cited on page 28).
- Zeng, Wenying, Yuelong Zhao, Kairi Ou and Wei Song. 2009. 'Research on cloud storage architecture and key technologies'. In *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human: ICIS '09*, 1044–48. Seoul, Korea, 24th–26th November. ISBN: 978-1-60558-710-3. doi:10.1145/1655925.1656114. (Cited on page 23).

Zikopoulos, Paul C., George Baklarz and Dan Scott. 2010. *Apache Derby – Off to the races: Includes details of IBM Cloudscape*. IBM Press. ISBN: 978-0-13-708017-5. (Cited on page 40).